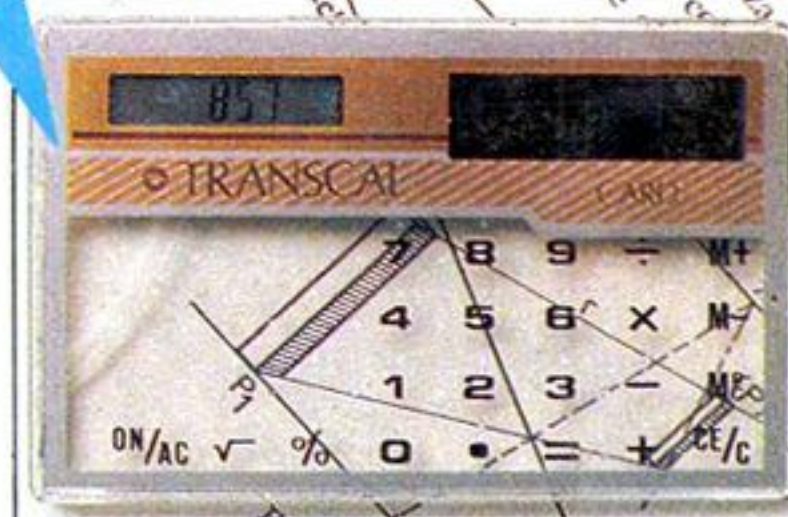
Anno XI - N.85 - 25 Giugno 1991
Sped. Abb. Post. Gr III/70 - CR- Distr.: Parrini

MATEMATICA

- Divisione infinita
- Oltre la calcolatrice
- Equazioni di 3º grado

Systems

AMIGA

Finalmente un Pascal vero

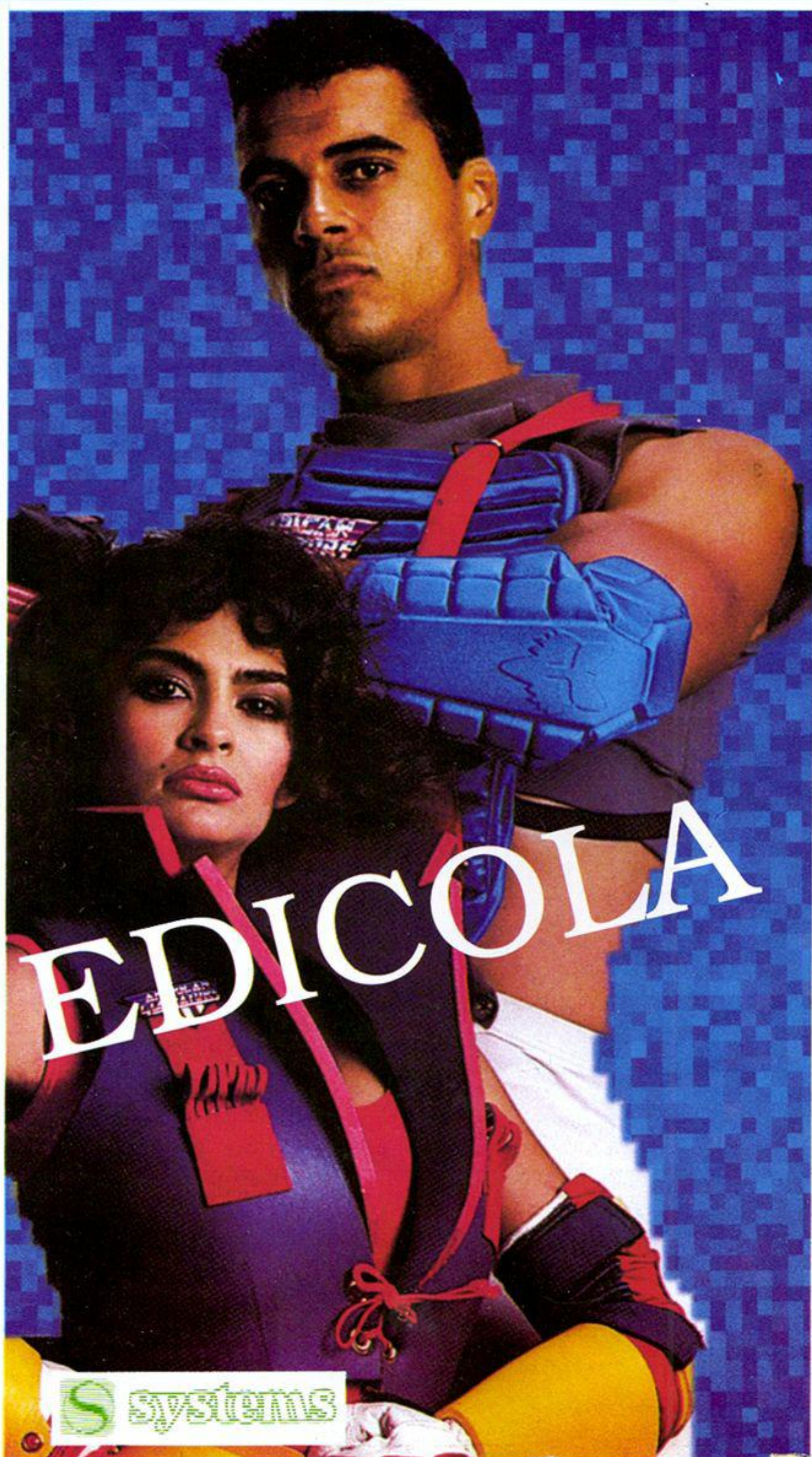
GIOCHI

- Come truccare un game
- Metti giù la briscola

GRAFICA

Immagini Hi-Res da 64 ad Amiga

E INOLTRE:

Grafica: plot in assembly 80X86 • Personalizzare un carattere • Una matrice per crittografare • Recensioni: Amiga Action Replay 2 e Musica in Microdeal • Amigazzetta 11 • File relativi in GFA Basic • Enciclopedia di routine • Il file Ansi.sys • Inchiesta tra i lettori • Quick Basic, C e Turbo Pascal

Sommario

A pagina 68 l'utile...

Inchiesta

Compila e spedisci!

Spazio 64

Amiga

Mondo Dos

Amiga + Ms- Dos

COMMODORE COMPUTER CLUB

Direttore: Alessandro de Simone
Coordinatore: Marco Miotti

Redazione / Collaboratori:
Davide Ardizzone - Claudio Baiocchi
Luigi Callegari - Umbero Colapicchioni
Donato De Luca - Carlo d'Ippolito
Valerio Ferri - Michele Maggi
Giancarlo Mariani - Domenico Pavone
Armando Sforzi - Dario Pistella
Fabio Sorgato- Valentino Spataro
Franco Rodella- Stefano Simonelli
Luca Viola

Direzione:
Via Mosè, 22 cap. 20090 OPERA (Mi)

Telefono 02 / 57.60.63.10
Fax 02 / 57.60.30.39
BBS 02 / 52.49.211

Pubblicità:
Leandro Nencioni (dir. vendite)
Via Mose', 22 20090 Opera (Mi)
tel. 02 / 55.60.63.10

Emilia Romagna:
Spazio E
P.zza Roosvelt, 4 cap. 40123 Bologna
Tel. 051 / 23.69.79

Toscana, Marche, Umbria
Mercurio s.r.l. Via Rodari, 9
S. G.nni Valdarno (Ar)
Tel. 055 / 94.74.44

Lazio, Campania
Spazio Nuovo
Via P. Foscari, 70
cap. 00139 Roma
tel. 06 / 81.09.679

Abbonamenti: Liliana Spina
Arretrati e s/w: Lucia Dominoni

Tariffe: Prezzo per copia L. 6000
Abbonamento annuo (11 fascicoli) L. 60000
Estero: L. 100000 - Indirizzare versamenti a:
Systems Editoriale Srl c/c 37952207 oppure
inviare comune assegno bancario non trasferibile
e barrato due volte a:
Systems Editoriale Srl (servizio arretrati)
Via Mose', 22
cap. 20090 OPERA (Mi)

Composizione: Systems Editoriale
La Litografica Srl Busto Arsizio (Va)

Registrazioni: Tribunale di Milano
n. 370 del 2/10/82

Direttore Responsabile: Michele Di Pisa

Spedizioni in abbonamento postale gruppo III.
Pubblicita' inferiore al 70%

Distributore: Parrini - Milano

Periodici Systems:
Amiga Club (disco ed. Germania) - Banca Oggi - Computer (quotidiano) - Computer Club - 64 Club (disco) - Computer Club (disco ed Germania) - Hospital Management - Nursing '90 - PC Club (disco ed. Germania) - Personal Computer - Jonathan - VR - Videoteca

Crescendo s'impara

Non tutti i nostri lettori ricordano le "scadenze" indicate, fino a qualche numero fa, sulle prime pagine della nostra rivista.

Mi riferisco alla prossima, inevitabile estromissione di ogni argomento legato al glorioso **C/64**, ormai lontano dal *target* degli utenti di informatica che, per hobby o professione, si rivolgono sempre più verso computer a 16/32 bit.

Sorvolando sull'eterno, "falso" dilemma (**Amiga** oppure **Ms - Dos**?) che ci vedrà ancora impegnati nel sedare pacifiche risse tra opposte fazioni, ci preme evidenziare un aspetto che, impensabile fino a poco tempo fa, inizia ad emergere in modo quasi sistematico.

Se, infatti, il nome Commodore rappresentava, in precedenza, un ben preciso ambiente (se non, addirittura, un vero e proprio "mondo" a parte), oggi come oggi le differenze tra i vari ambienti tendono ad assottigliarsi, fino quasi a scomparire.

Anzitutto, in campo Ms - Dos, non ha senso parlare di elaboratori e periferiche di sola marca Commodore. La totale compatibilità Ms - Dos di tutte le marche (e, quindi, Commodore in particolare), infatti, non giustifica l'esclusiva di un nome rispetto ad un altro.

Anche in ambiente software, inoltre, la commercializzazione di uno stesso pacchetto in più versioni (Ms - Dos, Amiga, Macintosh) impedisce di privilegiare un prodotto rispetto ad un altro.

La passione per l'informatica (e qui ci rivolgiamo ai nostri lettori) ed il costante abbassamento dei prezzi, infine, consente all'hobbysta evoluto di possedere più macchine insieme, non necessariamente tutte di una marca unica.

E' questo il motivo per cui il nome **Commodore** non compare più nella nostra testata, mutata in **"Computer Club"** e quindi priva di riferimenti ad un qualsivoglia nome commerciale.

Inutile dire che l'impegno richiesto è maggiore che nel passato; del resto, ampliando l'orizzonte, il viaggio da intraprendere richiede altre energie, da utilizzare nella continua crescita culturale che l'informatica, più che richiedere, esige.

Alessandro de Simone

LA VOSTRA POSTA

(a cura di A. de Simone)

Ernesto il censore

Si vede che ritenete che l'esposizione di natiche moltiplicate o di poppe da 5 Kg l'una possa guadagnarvi dei lettori, ma in questo modo voi offendete solo l'intelligenza di questi ultimi, che cercano nella rivista ciò che interessa veramente l'amatore di computer (senza parlare dell'offesa che viene portata di continuo alla donna, mostrata sempre e solo come oggetto di piacere). Seguo anche molte altre riviste di informatica, sia italiane che americane, ma nessuna di queste ritiene necessario ricorrere a mezzi così rozzi.

(Ernesto C. Roma)

Come è possibile notare dalle immagini pubblicate (qualità della carta permettendo) queste sono sempre e solo immagini digitalizzate che hanno, come scopo principale, quello di evidenziare il notevole progresso compiuto nel campo del "trattamento" di immagini, soprattutto grazie all'introduzione di hardware e software sempre più sofisticato.

Da che mondo è mondo, poi, la bellezza è sempre stata monopolio delle donne; quale migliore scelta, quindi, può essere fatta per evidenziare al massimo le qualità grafiche di una scheda o la possibilità di manipolazione di un'immagine?

Spesso, ed il lettore dovrà riconoscerlo, pubblichiamo anche immagini digitalizzate di quadri famosi, sempre allo scopo di evidenziare la notevole fedeltà delle sfumature di colore e di altre caratteristiche grafiche, proprie di immagini di elevata qualità.

Non è colpa nostra, poi, se chi decide di digitalizzare immagini fa cadere la sua scelta su particolari tondeggianti e rosei. Se il nostro lettore, comunque, dispone di scanner o di telecamera (e, ovviamente di un computer) può inviarci diapositive di Andreotti, di Marzullo o di altri insospettabili personaggi che possano comunque generare una limitata (e, ne siamo sicuri, involontaria) eccitazione sessuale; purché siano immagini, ripeto, tendenti ad evidenziare le caratteristiche della macchina, non a sottovalutarle.

Doppio foro

Effettuando un foro "gemello" di quello che rappresenta la protezione di un dischetto da 3.5 è possibile trasformare un 720 Kbyte in 1.44 megabyte?

(da alcune lettere)

La qualità di moderni supporti magnetici consente oggi (come ieri con il drive 1541 per il C/64) di praticare un foro in modo da far riconoscere, da un elaboratore Ms - Dos, il dischetto come di capacità elevata (parliamo di sistemi Ms - Dos, non di Amiga).

Stavolta, però, c'è un problema che non si presentava con i dischetti da 5.25, facilmente tagliabili con una comune taglierina.

L'involucro dei 3.5, infatti, è di materiale plastico ed obbliga ad usare un trapano per praticare il foro. Ciò comporta l'inevitabile generazione di truciolo di plastica che potrebbe insinuarsi all'interno dell'involucro stesso e generare vari danni in fase di utilizzo del dischetto.

Sono attualmente in vendita particolari tranciatori (piuttosto costosi) che, oltre ad evitare il problema del truciolo (troncano di netto il quadratino di plastica e ne consentono la rimozione) sono in grado di conferire un contorno preciso al foro.

Sull'affidabilità di un'operazione del genere, tuttavia, è bene avere qualche riserva: è infatti noto che, a volte, operando su **veri** dischetti da 1.4 megabyte sorgono problemi di compatibilità tra drive diversi.

Figuriamoci usando un semplice dischetto "potenziato" a 1.4 mega.

Più semplice di così...

Anch'io vorrei dire la mia sulla mini-sfida lanciata sul n. 81, nella quale si chiedeva di indicare una procedura per attivare un file batch dotato dello stesso nome di un comando Ms-Dos...

(Carmine Russo - Vitulazio)

E' infatti sufficiente *-continua il nostro lettore-* far precedere il nome del file dalla barra inclinata inversa (\), come ad esempio...

\date

...per attivare un file batch di nome `date.bat`. La particolare forma sintattica, pur se non molto nota, evita il ricorso a complicate procedure".

E bravo il nostro Carmine. E' proprio vero che non si finisce mai di imparare!

Bello e complesso

Vi invio la versione in Pascal del programma "Indovina indovinello" da voi pubblicata, in Basic, in un precedente fascicolo.

(Lorenzo Pollini - Grosseto)

Il programma inviato (senza, purtroppo, un preventivo accordo) è di notevole interesse, soprattutto per quanto riguarda la gestione dell'"**albero di ricerca**".

Le spiegazioni a corredo, tuttavia, sono troppo scarse e le routine inserite nel listato non sono facilmente interpretabili. La lunghezza del programma, infine, è eccessiva e

Il motivo per cui scrivo questa lettera (con l'ausilio del grande **ProWrite 2.0**) è presto detto: non mi piace come ultimamente Computer Club parli dell'Amiga. Tutto il peggio che c'era da dire è stato detto (nei pungenti editoriali del mega direttore, che probabilmente starà leggendo la mia lettera). Vorrei ora argomentare una serie di difese (o contrattacchi) nei riguardi di questo sottovalutato computer (o snobbato, che è poi lo stesso). Partiamo con la grafica.

L'Ms - Dos con la scheda SuperVGA riesce veramente ad umiliare l'Amiga (ho avuto modo di vedere immagini allo SMAU, spaventose), per non parlare della nuovissima XGA, di qualità paragonabile a quella dei videoregistratori. Bene, provate a contare chi possiede la SuperVGA e poi fatemelo sapere. Perfino le piccole emittenti televisive usano l'Amiga per i loro (molteplici) scopi, preferendola all'Ms - Dos per un'infinità di cose, non ultimo il prezzo (perché non avete indicato il prezzo della SuperVGA nella risposta al lettore che chiedeva un paragone tra Ms - Dos e Amiga?).

Un Ms - Dos dotato di semplice VGA non concorre minimamente con l'Amiga (ma bisogna sempre considerar che la VGA costa, e non poco, mentre la grafica dell'Amiga è "fornita" con il computer stesso),

in quanto il modo HAM ha la meglio (in ogni caso l'hires dell'Amiga è quasi simile ai risultati di una VGA 640 x 480 con 16 colori).

Oltretutto, l'unico limite dei 4096 colori verrà presto superato con schede che porteranno la palette dei colori a 16 milioni (attendiamo il Video Toaster).

Per quanto riguarda il CAD, anche il tanto conclamato Autocad non regge con programmi analoghi sull'Amiga (soprattutto DynaCadd).

Una voce

Non tiro nemmeno in ballo Blitter e Copper, perché si potrebbe parlare una giornata intera sui vantaggi che comportano. Discorso grafica chiuso, nettamente a favore dell'Amiga. Per quanto riguarda il suono, la scheda Sound Blaster costa 335 mila lire e non è quasi mai utilizzata decentemente dai giochi; in ogni caso, per le Work Station musicali non viene utilizzato l'Ms - Dos bensì l'Amiga (e, sporadicamente, l'Atari ST).

Non avete ancora parlato del nuovo sistema operativo (chissà perché). Comunque il Workbench ha un aspetto eccellente (soprattutto se si definisce un retino come sfondo, evitando di usare quelli pre-definiti), e non mi si venga a dire che Windows per Ms - Dos è efficiente come il WB 2.0.

Oltretutto l'AmigaDos (fino alla release 1.3) è stato riscritto, col risultato di avere comandi più veloci ed efficienti. Non riesco a capire come mai parliate del Quick Basic e del Turbo Pascal ed ignoriate i linguaggi Basic sull'Amiga.

Possiedo il superbo GFABasic. Ebbene: questo concorre senza problemi con i tanto conclamati QuickBasic e T. Pascal (non c'è una sola performance, tra quelle elencate da voi stessi, che non sia posseduta anche dal primo). Sono riuscito a trascrivere la routine del numero 82, "Barra" a pagina 20 e la velocità ottenibile in GFABasic compilato parla da sè: con gli stessi parametri passati al QuickBasic impiegava 12 secondi interpretato e 10 compilato! Però il QuickBasic funzionava su un computer a 10 Mhz con coprocessore matematico: in conclusione il GFABasic è MOLTO più veloce del QuickBasic (perché non avete pubblicato i tempi del T. Pascal?). Ritengo di essere in grado di tradurre tutte le routine che pubblicate in QuickBasic, T. Pascal e AmigaBasic (a proposito: sarò forse l'unico al mondo a non conoscerlo minimamente, perché son "partito" subito con GFABasic). L'unico mio ostacolo è, appunto, quello di conoscere solo superficialmente i due linguaggi citati; nonostante ciò sono sempre riuscito nella conversione dei programmi.

Inoltre parlate come se l'unico linguaggio decente per Amiga sia il C: non è vero.

Non parliamo poi dell'Amos, con cui si ottengono dei risultati (rigorosamente in campo grafico-musicale) che in campo Ms - Dos si otterrebbero a malapena in assembly (sempre tenendo conto della grafica Dossiana, scarsuccia), e in campo Amiga supera spesso quelli del C.

Mi piacerebbe molto quindi che parlaste del GFABasic (se chiedete alla SoftMail o alla Eurosoft stessa, verificherete che è un linguaggio MOLTO richiesto, anche se gli Amighi utilizzano il loro computer solo per giocare e tra parentesi li impiccherei tutti), nonché del nuovo sistema operativo (che conosco peraltro piuttosto bene).

Ho concluso la mia lettera. Spero proprio che vorrete rispondere ai numerosi "massi" che ho lanciato.

La mia lettera non vuole assolutamente essere polemica (come è stato detto per la mia precedente, titolandomi come "criticatutto"). Io vi dico queste cose semplicemente perché vorrei la rivista sempre migliore (a proposito: in questi ultimi mesi siete MOLTO migliorati).

Spero abbiate la cortesia di pubblicare il mio indirizzo sulla rivista, per un eventuale scambio di idee con i lettori, soprattutto se appartenenti al "partito" Ms - Dos.

Mauro Bossetti
via Buozzi 2/a
20090 Trezzano s/N (Mi)

richiederebbe un numero elevato di pagine della rivista.

E' tuttavia evidente la bravura dimostrata dal nostro lettore che, se lo ritiene opportuno, può contattarci per concordare insieme un articolo di sicuro interesse.

Bello e semplice

Ho apportato alcune modifiche al programma "Barre" da voi pubblicato nella rubrica "Enciclopedia di routine "del N. 82 e mi piacerebbe vederlo pubblicato.
(Carmelo R.)

La modifica di cui parla il nostro lettore consente di ottenere le stesse cose, proposte a suo tempo, ma in bassa risoluzione invece che in Hi-res.

Dal momento che, oggi, l'utente medio Ms - Dos dispone sempre di un monitor grafico, non si giustificherebbe la pubblicazione di un listato che, in effetti, rappresenta una "decurtazione" delle possibilità offerte da un software divulgato in precedenza sulle nostre pagine.

I PICCOLI COMMODORE SONO CRESCIUTI

Una sfida al mese

W i triangoli, purché rettangoli

Disegnare un triangolo sembra la cosa più facile di questo mondo; vediamo se è proprio vero...

L'area del quadrato costruito sull'ipotenusa è eguale alla somma delle aree dei quadrati costruiti sui due cateti.

Ipse (= Pitagora) *dixit;* e c'è da credergli, se da alcune centinaia di anni il celebre teorema continua a corrugare le fronti di milioni di studentelli in tenera età.

Chiunque è in grado di disegnare con una certa precisione, magari a mano libera, le quattro figure (triangolo e tre quadrati) geometriche che dimostrano il teorema.

Ma, "passando" il compito al computer, è davvero così semplice realizzare quadrati e triangolo a suon di istruzioni grafiche? Vediamo, insieme, di chiarire il problema:

Qualunque schermo, rettangolare, offre una limitata risoluzione (e questo, di per sè, può esser motivo di sofferenza). Inoltre, a causa di particolari dimensioni del triangolo, uno dei quadrati può eccedere i limiti dello schermo. Infine, tenendo conto del potenziale utente (studente di scuola media), bisogna realizzare un'interfaccia utente semplice ed immediata.

La sfida

In pratica la sfida di questo mese consiste nel realizzare un software in grado di dimostrare il teorema di Pitagora; a tale scopo l'utilizzatore, dopo aver lanciato il programma (scritto, inutile dirlo, con qualsiasi linguaggio e con qualsiasi computer, purché non obsoleto) deve:

* Veder apparire un triangolo rettangolo equilatero "di partenza", di minime dimensioni (vedi ABC di figura), completo di quadrati costruiti sui suoi lati, sul quale effettuerà i vari esperimenti. In alto (e/o in basso) sul video devono apparire le misure dei tre lati e delle tre aree che dimostrano la validità del teorema.

* Subito dopo, utilizzando possibilmente il **mouse**, indicare il lato del triangolo, orizzontale o verticale, da allungare o accorciare.

* Contemporaneamente (cioè **in tempo reale**), le quattro figure geometriche, e le quote numeriche che compaiono sul video, devono adeguarsi alla nuova impostazione.

* Nel caso vengano impostate misure inadeguate, il software deve provvedere automaticamente a spostare il punto B in modo da far rientrare nell'area video, per quanto possibile, le figure.

Schema dell'output su video del programma da realizzare

Se vi sembra troppo facile, bè, mettetevi ugualmente all'opera: tra quelli inviati verrà pubblicato, infatti, il programma più breve che è in grado di rispondere adeguatamente alle esigenze segnalate.

Il premio in palio, che verrà consegnato all'autore del lavoro selezionato, consisterà in un **funzionale appendiabiti di stile post-moderno** (sarà cura del vincitore comunicare, indicandola in centimetri, la misura del **chiodo** che intende ricevere in omaggio).

Invece dell'elemento di arredo, ovviamente, sarà anche possibile richiedere pubblicazioni della Systems editoriale.

Come partecipare

Nelle sfide mensili lanciate dalla nostra Testata viene sempre data precedenza ai programmi, scritti in qualsiasi linguaggio, in grado di girare su **Amiga** oppure computer **Ms-Dos compatibili**. I listati devono essere strutturati in modo chiaro, allo scopo di favorire eventuali modifiche da parte di lettori che intendano sofisticare la procedura. Listati ed articolo esplicativo devono tassativamente essere inviati su **disco** (qualsiasi formato) oppure, via modem, alla nostra **BBS** (tel. **02/57.60.52.11**). In ogni caso, onde evitare invio di materiale inadeguato alla pubblicazione, si consiglia di telefonare in redazione (Tel. **02/57.60.63.10**) per concordare l'inoltro del lavoro svolto.

CRESCI
ANCHE TU
COL NUOVO

COMPUTER CLUB

di Dario Pistella

Come "passare" immagini grafiche dal C/64 all'Amiga

Una procedura che permette di "trasportare" su Amiga tutte le immagini in alta risoluzione che avevate memorizzato con il C/64

Il semplice programma proposto in queste pagine permette ad Amiga di interpretare e visualizzare schermate grafiche provenienti dal vecchio **C/64**.

Il formato utilizzato dai due computer risulta diverso, dal momento che Amiga ricorre ad un "codice" che in questi ultimi anni sta avendo una larghissima diffusione, l'**IFF** (**I**nterchange **F**ile **F**ormat).

Proposto come file universale per la memorizzazione dei dati dalla **Electronic Art** nel 1985, venne accettato dalle maggiori compagnie produttrici di software, vista la versatilità e le molteplici funzioni. Di solito la somma di 100 dollari per un programma che consente solo di disegnare è considerata una cifra alta; ma se il software permette anche di realizzare titoli video, animazioni e grafica di ogni genere, certamente è un investimento più equo e completo.

Purtroppo parecchi programmi di disegno per personal computer piuttosto diffusi e popolari non permettono l'interscambio dei file, cosicché un file creato con il programma "A" non può essere utilizzato con il programma "B". Il Commodore 64, ad esempio, è una vittima di questo tipo di incompatibilità.

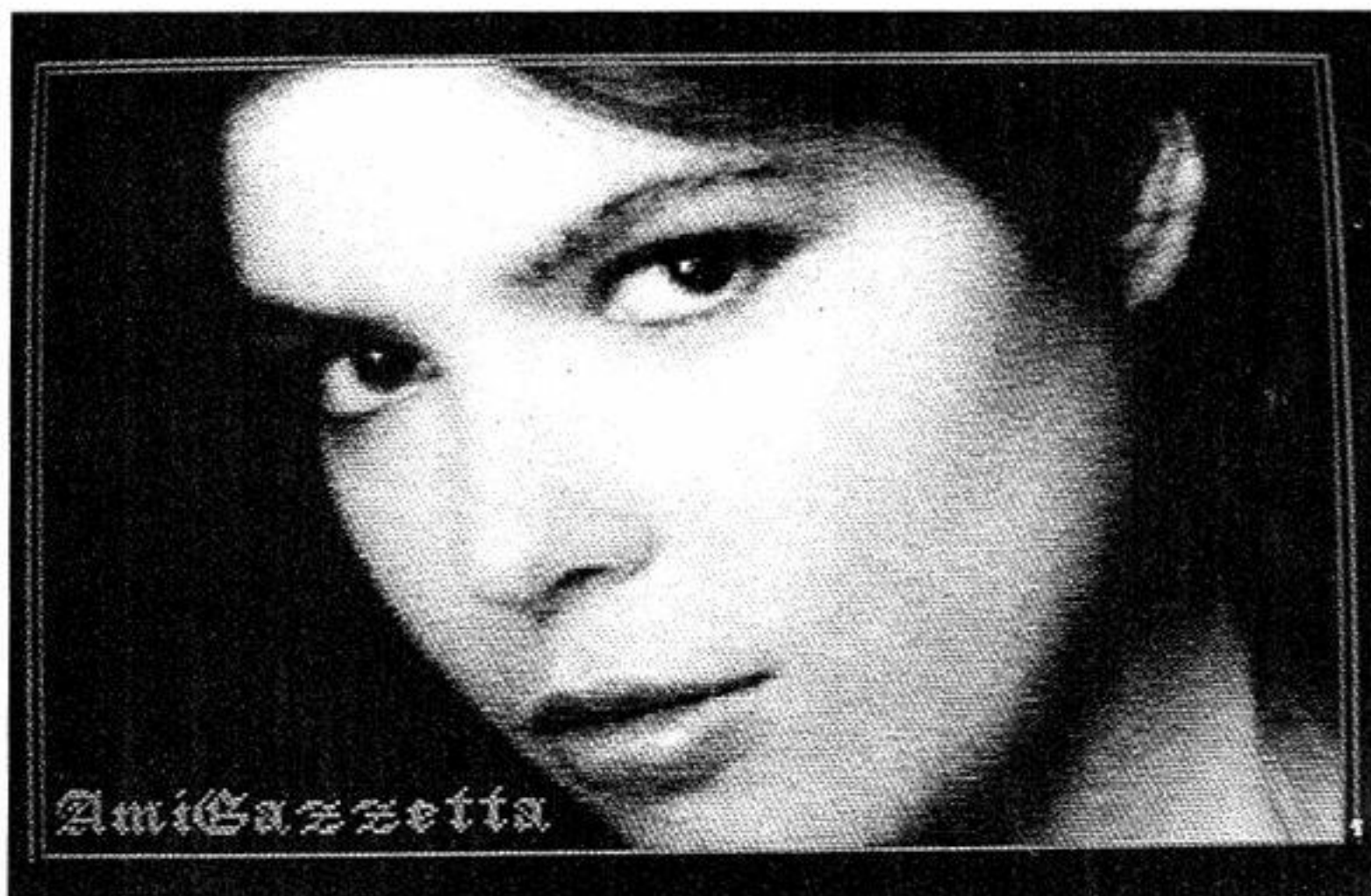
Chunk e Form

Nello standard IFF i dati che svolgono la stessa funzione sono raggruppati in un unico blocco, definito **Chunk**.

Un file IFF è composto da più Chunk. Tutti i chunk sono riuniti in un'unica struttura, detta **Form**. Un programma Amiga verifica il tipo di ogni *chunk* controllando l'identificatore di ciascun *form*, che contiene le informazioni riguardanti la lunghezza ed il tipo di file.

Bitmap

Un file grafico IFF è composto, come minimo, da tre chunk. Il primo contiene i **dati di controllo**, come ad esempio la *larghezza* e la *lunghezza* del disegno. Il secondo chunk contiene i valori **RGB** per i colori. Il terzo contiene i **bit** che compongono il disegno vero e proprio.

L'insieme di bit che costituisce un disegno è chiamato **bitmap**. Il bitmap può utilizzare più chunk come, ad esempio, per i dati che realizzano una *finta animazione* con i colori (probabilmente si è avuto modo di vedere uno di quei giochi che utilizzano il cambio ciclico di un colore, per dare l'idea del movimento), oppure le modalità operative del video, oppure per le coordinate associate al punto un cui un segmento grafico viene visualizzato sullo schermo.

```
REM IFFER (Convertitore di schermate Hi - Res da C/64 ad Amiga)
REM by
REM Dario Pistella

DIM v (39, 7)
INPUT "File sorgente ";sorg$
INPUT "File oggetto ";ogg$
OPEN "df0:" + sorg$ FOR INPUT AS 1
OPEN "df0:" + ogg$ FOR OUTPUT AS 2
PALETTE 1, 1, 1, .13: GOSUB IFF

BMHeader:
   Ascissa = 320: Ordinata = 200
   SCREEN 1, Ascissa, Ordinata, 5, 1: WINDOW 2, nomedue$, , 0, 1
   Locazione = PEEKL (WINDOW (8) + 4) + 8
    Pianolocazione = PEEKL (Locazione)

Bodymap:
 FOR Linee = 0 TO 24
  FOR ottoperotto = 0 TO 39
  FOR bytevert = 0 TO 7

   valore = ASC (INPUT$ (1, #1))
   POKE Pianolocazione + ottoperotto + bytevert
* 40 + 320 * Linee, valore
   v (ottoperotto, bytevert) = valore

  NEXT bytevert
  NEXT ottoperotto

REM conversione
   FOR bytev = 0 TO 7
   FOR cic = 1 TO 5
    FOR oxo = 0 TO 39
 PRINT#2, CHR$ (v (oxo, bytev));
    NEXT oxo
   NEXT cic
   NEXT bytev
 NEXT Linee
CLOSE 2: CLOSE 1: END

IFF:
FOR k = 1 TO 186: READ a: ck = ck + a
PRINT#2, CHR$ (a);: NEXT
IF ck = 14999 THEN RETURN
PRINT "Errore nei data.": STOP
DATA 70, 79, 82, 77: REM FORM
DATA 0, 0, 156, 242: REM 40178 bytes
DATA 73, 76, 66, 77: REM ILBM
DATA 66, 77, 72, 68: REM BMHD
DATA 0, 0, 0, 20: REM 20 bytes
DATA 1, 64, 0, 200, 0, 0, 0, 0

DATA 5: REM Bitplane
DATA 0, 0, 0, 0, 0, 10, 11, 1, 64, 0, 200
DATA 67, 77, 65, 80: REM CMAP
DATA 0, 0, 0, 96: REM 96 bytes
```

Identificatori

Gli identificatori di chunk e di form sono costituiti da **quattro** byte.

Questi compongono una parola di codice riconosciuta dalla routine di lettura IFF, e sono seguiti da *altri quattro byte* che mantengono le informazioni relative alla quantità di dati del disegno.

Form e Ilbm

I primi quattro caratteri del file contengono la parola **Form**. I successivi quattro byte contengono la *lunghezza* della form, memorizzata come numero a 32 bit.

Di seguito vi è un identificatore che specifica il tipo di file, che può essere, come nel caso del file generato dal programma di queste pagine, di tipo **ILBM** (**Inter**Leaved **BitM**ap), che indica come il bitmap sia combinato con altri dati in un'unica form.

Bmhd e Cmap

Nel primo chunk seguono altre informazioni: quattro byte che contengono la parola **BMHD** (**BitM**ap **Head**er) ed *altri quattro* che forniscono la lunghezza del chunk (20 byte, nel nostro caso).

In questo chunk sono precisate le dimensioni del disegno ed il numero di **Bitplane**.

Proseguendo nell'analisi del formato IFF si trova l'identificatore **CMAP** (**C**olor **MAP**, mappa dei colori) seguito dalla sua *dimensione*. Questo chunk contiene i dati per i valori *RGB* dei vari colori utilizzati nella *Palette*. Le tre tonalità red, green e blue usano fino ad un byte ciascuna.

Body

Il successivo chunk è quello contrassegnato dall'identificatore **Body**, che costituisce il bitmap del disegno.

I dati contenuti al suo interno seguono lo standard IFF: le linee che costituiscono i vari bitplane sono accatastate sequenzialmente nel file (la prima linea del primo bitplane, seguita dalla seconda linea del secondo bitplane, etc.).

```
DATA 64, 64, 64, 240, 112, 112, 144, 144, 112, 144, 176, 144,
   112, 128, 112, 240, 48, 80, 128
DATA 144, 144, 208, 144, 128, 192, 128, 128, 208, 160, 128,
   128, 128, 112, 48, 48, 48, 0, 112
DATA 240, 80, 80, 80, 112, 96, 96, 160, 128, 128, 80, 80, 80,
   80, 80, 80, 192, 160, 144
DATA 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 192, 160,
   144, 96, 112, 128, 144, 112, 96
DATA 192, 128, 112, 128, 96, 80, 160, 96, 96, 48, 80, 112, 128,
   128, 128, 160, 192, 160
DATA 67, 65, 77, 71, 0, 0, 0, 4, 0, 0, 64, 0, 67, 67, 82, 84,
   0, 0, 0, 14
DATA 0, 1, 16, 22, 0, 0, 0, 0, 0, 1, 61, 224, 0, 0
DATA 66, 79, 68, 89: REM BODY
DATA 0, 0, 156, 64: REM 40000 bytes
END
```

Interpretazione dei dati

Il problema fondamentale è dato dal fatto che il C/64 non utilizza questo tipo di formato ed inoltre i dati che nell'IFF costituiscono il Body sono disposti in modo diverso dai vari programmi grafici del C/64.

Da Print Shop ad Amiga

Il programma presentato prende in considerazione i file generati dal ben noto programma **Print Shop** (ci riferiamo all'"ambiente" C/64), i cui file generati constano di **8000 byte** suddivisi come segue: 25 gruppi di byte a loro volta suddivisi in 40 gruppi di 8 byte. Ma procediamo con ordine: nel file memorizzato con il C/64 vengono registrati, sequenzialmente, gli 8 byte che costituiscono le 8 linee di 8 pixel della prima cella in alto a sinistra sullo schermo, seguiti da altri otto byte che costituiscono le 8 linee di 8 pixel della casella a fianco, e così via.

Dopo 40 gruppi di 8 byte si avranno quindi tutti i dati relativi alla prima linea di 8 x 8 pixel. Questo si ripete per 25 volte allo scopo di ottenere, alla fine, il disegno di 320 x 200 pixel.

Nel Chunk Body dei file IFF, invece, i dati sono disposti come segue: 40 byte che rappresentano il disegno della prima linea di 1 pixel, seguiti da altri gruppi di quaranta byte (a seconda del numero di bitplane) sempre relativi alla prima linea; questi sono seguiti, a loro volta, da altri 40 byte relativi ai punti della seconda linea di 1 pixel, anche questi seguiti da altri gruppi di 40 byte a seconda del numero di bitplane. La sequenza viene ripetuta per 200 volte fino a definire un intero disegno di 320 x 200.

Conversione

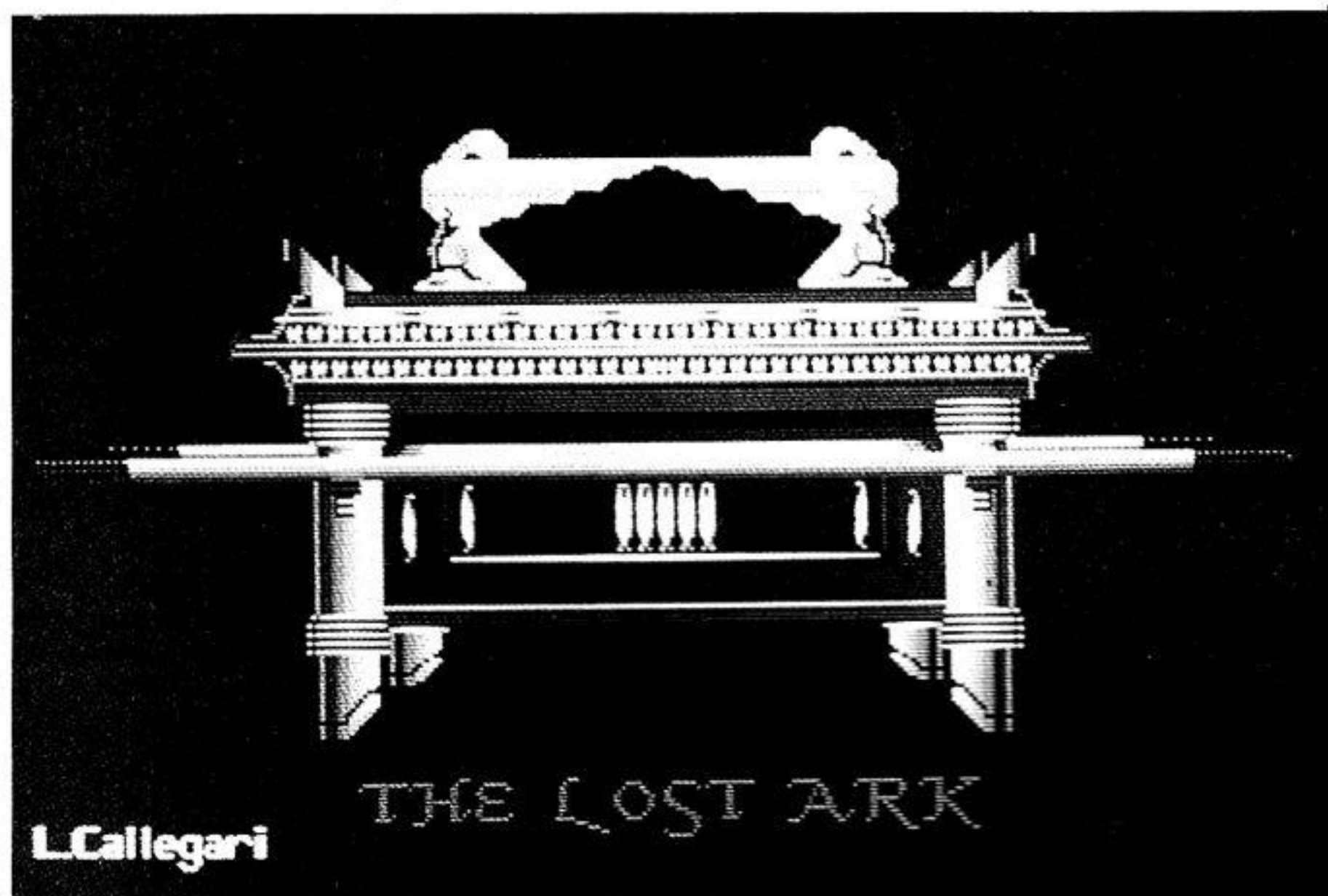
Il programma presentato in queste pagine (e che gira, ovviamente, su Amiga) si occupa della "traduzione" di schermate del C/64 (in formato **Print Shop**) in formato IFF.

Dapprima è necessario trasferire il file contenente i dati della schermata da C/64 ad Amiga (servendosi di utility del tipo **Dos2Dos** oppure **64 Emulator** oppure utilizzando un cavetto **Rs-232** collegato tra i due computer); quindi sarà sufficiente lanciare il programma.

Dopo aver richiesto i nomi dei due file, il primo da interpretare (proveniente dal C/64) ed il secondo da registrare (da "trattare", in seguito, con Amiga), scrive in quest'ultimo i dati fissi riguardanti BMHD e CMAP (320 x 200, bitplane = 1, colori fissi).

Quindi legge il file sorgente visualizzandolo, mettendo ogni singolo byte nella variabile V (colonna, linea della casella).

Dopo aver letto 320 byte (cioè 40 colonne x 8 linee), provvede a registrare il file nel formato IFF.

A programma terminato, sul dischetto sarà presente il nuovo file IFF contente il disegno rielaborato, da manipolare, eventualmente, servendosi di un qualsiasi programma grafico in grado di elaborare immagini IFF.

di Giancarlo Mariani

Introduzione alla grafica con un mini listato 80X86

Di solito si pensa che un programma grafico in assembler debba esser lungo e complesso; provvediamo a sfatare questo luogo comune

Nelle scorse chiacchierate abbiamo visto alcune istruzioni che permettono di confrontare valori (**CMP**) tra loro, quindi la modifica di tali valori (**INC / DEC**) e la possibilità di prendere decisioni a seconda dei risultati ottenuti (**JC / JNC / JZ / JNZ**).

Inoltre abbiamo parlato anche dell'istruzione **INT**, che serve per richiamare funzioni particolari del BIOS del PC.

In queste pagine non vogliamo parlare di nuove istruzioni assembler, ma di **altre interessanti applicazioni** dell'istruzione INT, con un programmino di esempio che fa uso di parecchie delle istruzioni apprese sinora.

La volta scorsa

Riportiamo, di seguito, una breve descrizione dell'istruzione INT, per chi non avesse sottomano il fascicolo precedente (ma che aspettate a procurarvi gli arretrati?).

L'istruzione serve per richiamare alcune delle routines presenti nel BIOS (cioè le **Rom** montate nel computer dal fabbricante del computer) o nel DOS (parte software caricata automaticamente al momento del Boot di sistema) oppure delle routines costruite dal programmatore per svolgere determinati compiti.

Tutto ciò al fine di sfruttare, in propri programmi, routines già costruite e residenti (in un modo o nell'altro) nel vostro PC.

INT si usa...

INT Interrupt

...in cui *INT* è il codice mnemonico dell'istruzione, abbreviazione dell'inglese INTerrupt, ossia interruzione.

Interrupt è il numero dell'interruzione da richiamare, e può essere **solo** un valore immediato.

I vari parametri da passare all'istruzione vengono immessi nei registri (**AX, BX, CX**, ecc.).

Le funzioni di cui ci occuperemo stavolta sono dedicate, principalmente, ad una semplice gestione della grafica.

INT 10h, Funzione 0

Impostazione modo video; questa funzione viene utilizzata per decidere in che

Modo(HEX),	Tipo(T/G),	Colori,	Scheda,	Formato video
00,	T,	2,	COLORE,	40x25
01,	T,	16,	COLORE,	40x25
02,	T,	2,	***,	80x25
03,	T,	16,	COLORE,	80x25
04,	G,	4,	COLORE,	320x200
05,	G,	4,	COLORE,	320x200
06,	G,	2,	COLORE,	640x200
07,	T,	4,	MDA/EGA,	80x25
08,	G,	16,	PCjr,	160x200
09,	G,	16,	PCjr,	320x200
0A,	G,	4,	PCjr,	640x200
0D,	G,	16,	EGA,	320x200
0E,	G,	16,	EGA,	640x200
0F,	G,	2,	EGA,	640x350
10,	G,	16,	EGA,	640x350
11,	G,	2,	VGA,	640x480
12,	G,	16,	VGA,	640x480
13,	G,	256,	VGA,	320x200

Tabella 1. I formati del video e le loro caratteristiche

modo predisporre il video (testo o grafica). I parametri da passare alla funzione sono:

ah = 0 (Funzione 0)
al = Modo

Il **Modo** deve essere selezionato tramite la tabella 1. In questa, *Modo* rappresenta il valore esadecimale da inserire nel registro **Al** prima di richiamare l'interrupt; *Tipo* è **T** (testo) oppure **G** (grafica); *Colori* è il numero di colori utilizzabile in quel modo; *Scheda* è la scheda video necessaria per utilizzare il modo; *Formato video* è la dimensione del video, espressa in **pixel** (per i modi grafici) oppure in **caratteri** (per i modi testo).

Per esempio, per imporre la grafica 640 x 200 monocromatica, bisognerà programmare:

```
mov ah, 0 ;Funzione 0
mov al, 6 ;Modo 6
int 10h   ;Esegue la funzione
```

Si rammenta che il modo testo normale del video, per uno schermo 80 x 25, è il modo **3**. In caso di monitor **RGB**, non c'è differenza tra i modi **0** e **1** oppure tra i modi **2** e **3**. In caso di scheda **EGA**, i modi consentiti variano solo da **0** a **6** e da **0Dh** a **10h**.

Se il bit più alto del registro **Al** è posto a **1**, il display **non** viene cancellato quando si entra in un modo grafico. Questo permette di creare effetti interessanti, mescolando, sul video, caratteri di dimensioni differenti.

COLORE,	DESCRIZIONE
0,	Nero
1,	Blu
2,	Verde
3,	Azzurro
4,	Rosso
5,	Porpora
6,	Marrone
7,	Bianco
8,	Grigio
9,	Blu brillante
10,	Verde brillante
11,	Azzurro brill.
12,	Rosso brillante
13,	Porpora brill.
14,	Giallo
15,	Bianco brillante

Tabella 2. I codici dei colori

```
; DRAW.ASM:    Disegna in grafica sullo schermo
; tramite i tasti W, A, D, X
;              C cambia colore, ESC finisce
;              NB : TENERE CAPS-LOCK INSERITO!
cseg           SEGMENT PARA PUBLIC 'CODE'
               org 100h
               ASSUME cs:cseg, ds:cseg, ss:cseg, es:cseg

Start:
               mov  AX, CS
               mov  DS, AX             ;DS = CS
               mov  ES, AX             ;ES = CS

               mov  Xpos, 100     ;Posizione X, Y iniziale = 100, 100
               mov  Ypos, 100
               mov  Colore, 02h        ;Comincia con colore 01h
                                       ;(Sfondo nero, pixel verde)

               mov  Vmode, 10h         ;Modo grafico iniziale
               call VideoMode          ;Mette in grafica

AltroPixel:
               mov  ah, 0ch       ;INT 10h, funzione 0C: Disegna pixel
               mov  al, Colore    ;Colore del pixel
               mov  cx, Xpos      ;Coordinate X, Y
               mov  dx, Ypos
               int  10h           ;Richiama interrupt

               mov  ah, 8         ;Accetta un carattere da tastiera
               int  21h           ;Richiamo int. DOS (Carattere in al)

               cmp  al, 'W'            ;Tasto "W"?
               jnz  NoW                ;No : avanti
               dec  Ypos               ;Si : Decrementa posizione Y
               jmp  short AltroPixel   ;Riprende il ciclo

NoW:           cmp  al, 'X'            ;Tasto "X"?
               jnz  NoX                ;No : avanti
               inc  Ypos               ;Si : Incrementa posizione Y
               jmp  short AltroPixel   ;Riprende il ciclo

NoX:           cmp  al, 'A'            ;Tasto "A"?
               jnz  NoA                ;No : avanti
               dec  Xpos               ;Si : Decrementa posizione X
               jmp  short AltroPixel   ;Riprende il ciclo

NoA:           cmp  al, 'D'            ;Tasto "D"?
               jnz  NoD                ;No : avanti
               inc  Xpos               ;Si : Incrementa posizione X
               jmp  short AltroPixel   ;Riprende il ciclo

NoD:           cmp  al, 'C'            ;Tasto "C"?
               jnz  NoC                ;No : avanti
               cmp  Colore, 15         ;Colore = 15?
               jnz  NoCol15            ;No : salta
               mov  Colore, 0          ;Si : Azzera colore
               jmp  short AltroPixel   ;Riprende il ciclo
NoCol15:       inc  Colore             ;Incrementa colore
               jmp  short AltroPixel   ;Riprende il ciclo

NoC:           cmp  al, 27             ;Tasto ESC?
               jnz  AltroPixel         ;No : riprende il ciclo
               mov  Vmode, 3           ;Si : rimette in modo testo
```

INT 10h, Funzione 0Ch

Questa funzione è usata per disegnare un pixel (ossia un singolo punto) sullo schermo grafico. Ovviamente funzionerà solo se è stato prima selezionato uno dei modi grafici tramite INT 10h funzione 0, come descritto prima. I parametri da passare alla funzione sono:

```
ah = 0Ch    (Funzione 0C)
al = Colore del pixel
```

cx = Coordinata X
dx = Coordinata Y

Il **Colore** deve essere selezionato tramite la tabella 2.

Ovviamente tutti i colori funzioneranno solo sulle schede video e nei modi video che permettono il settaggio imposto (cioè in accordo con il numero di colori permessi, per ogni modo video, come nella tabella 2).

Le coordinate del pixel, come intuitivo, devono rientrare nei limiti leciti per il modo video selezionato. Per accendere, ad esempio, il pixel alla posizione **(x, y) 130, 100** dello schermo, bisognerà scrivere le seguenti istruzioni:

```
mov ah, 0Ch ;Funzione 0C
mov al, 4   ;Colore ROSSO
mov cx, 130 ;Coordinate
mov dx, 100
int 10h     ;Esegue la funzione
```

Se le coordinate vanno "fuori" dallo schermo, ossia dai limiti permessi da quel modo video, il pixel non sarà visibile.

Nei modi video che non offrono 16 colori, i valori da inserire nel parametro Colore (reg. Al) potranno variare tra 0 ed il numero di colori **-1**.

Se si setta a 1 il bit 7 del registro Al, il pixel verrà disegnato eseguendo un Or esclusivo (**Xor**) con il precedente contenuto della locazione di schermo.

INT 21h, Funzione 08h

Input carattere senza eco. Questa funzione viene utilizzata per inserire un carattere da tastiera, senza produrne l'eco sul video. I parametri da passare alla funzione sono:

```
ah = 08h    (Funzione 8)
```

La funzione restituisce in Al il codice (ASCII) del carattere introdotto da tastiera.

Per leggere, ad esempio, un carattere il programma assembler da digitare sarà il seguente:

```
mov ah, 08h ;Funzione 8
int 21h     ;Esegue la funzione
; al = codice caratt. digitato
mov Caratt, al ;Salva in
            ;"Caratt" il carattere
            ;introdotto
```

La funzione arresta l'esecuzione del programma fino a quando non viene introdotto un carattere. Se si premono i tasti Ctrl-c viene prodotto un **Int 23h** ed il programma in esecuzione termina.

INT 21h, Funzione 4Ch

Termina con codice di ritorno. Questa funzione permette di interrompere il programma in esecuzione e di ritornare al DOS o al programma chiamante. I parametri da passare alla funzione sono i seguenti:

```
ah = 4Ch     (Funzione 4C)
al = Codice di ritorno (Normalm. 0)
```

Il codice di ritorno può essere interpretato in procedure Batch con istruzioni del tipo If Errorlevel e quindi può essere esaminato per stabilire se un programma è andato a buon fine oppure no. La funzione si usa nel seguente modo:

```
mov ah, 4Ch ;Funzione 4C
mov al, 0   ;Codice di ritorno
            ;(Normalmente 0)
int 21h     ;Ritorna al prog.
            ;chiamante
```

Nell'uso dell'istruzione INT bisogna fare moltissima attenzione innanzitutto a richiamare interrupt "esistenti" ed inoltre a passare correttamente i parametri a questi ultimi. Se si richiama un interrupt non esistente, oppure si passano parametri errati, il PC potrebbe inchiodarsi, costringendo quindi a resettare.

```
            call VideoMode

            mov  ah, 4Ch            ;Ritorna al DOS
            mov  al, 0
            int  21h

;Subroutine VideoMode: Setta il modo del video
VideoMode:
            mov  ah, 0          ;INT 10h funz. 0 : Set video mode
            mov  al, Vmode      ;Modo
            int  10h            ;Richiama interrupt
            ret                 ;Ritorno al programma principale

; Dati usati dal programma
            Xpos   DW 0             ;Posizione X
            Ypos   DW 0             ;Posizione Y
            Colore DB 0             ;Colore del pixel
            Vmode  DB 0             ;Modo grafico

cseg        ENDS
            END  Start
```

Un esempio completo

Il programma che proponiamo questo mese cancella lo schermo ed impone il modo grafico, disegnando un pixel verde alla posizione 100, 100 dello schermo. Quindi si mette in attesa di un tasto, e a seconda di questo, prende le seguenti decisioni:

Tasto 'W' : Muove in alto e disegna un pixel
Tasto 'X' : Muove in basso e disegna un pixel
Tasto 'D' : Muove a destra e disegna un pixel
Tasto 'A' : Muove a sinistra e disegna un pixel
Tasto 'C' : Cambia colore
Tasto ESC : Torna al DOS

In pratica il programma consente di creare disegni perché, muovendo il cursore tramite i tasti **W**, **X**, **A**, **D**, si realizza una scia di pixel colorati. Il tasto **C** consente di "spazzolare" i colori da **0** a **15**. Ogni volta che si preme C, il colore verrà incrementato.

Premendo il tasto **Esc** il video torna in modo testo, e si ritornerà al DOS.

Il listato

Il programma assembler è pubblicato nell'apposito riquadro; esaminiamolo.

All'inizio ci sono le "solite" inizializzazioni dei registri di segmento **DS** ed **ES**, di cui ci siamo occupati in precedenza. Le successive 3 istruzioni (`mov...`) inizializzano le coordinate di partenza X, Y (100, 100) ed il colore verde (3).

L'istruzione "`Call VideoMode`" attiva la subroutine omonima per settare il video nel modo specificato dalla variabile **Vmode** (in questo caso, 10h = Grafica 16 colori 640 x 350, scheda EGA).

Il lettore dovrà ovviamente utilizzare un modo video compatibile con la sua scheda grafica.

Il successivo passo richiama l'interrupt **10h, funzione 0Ch**, che traccia il pixel alle coordinate **Xpos** e **Ypos** nel **Colore**.

Una volta disegnato il pixel, tramite l'**interrupt 21h funzione 8** si aspetta la pressione di un tasto e questo viene quindi memorizzato nel registro Al.

I successivi blocchi consentono di controllare il carattere premuto: CMP Al,.. JNZ consentono di ignorare il tasto non specificato in CMP (ricordiamo che JNZ salta se il risultato è diverso da quello specificato). I vari INC / DEC servono per aggiornare la posizione del cursore a seconda del tasto premuto; quindi JMP SHORT torna ad eseguire il ciclo.

In caso di pressione del tasto C, viene controllata la locazione Colore: se è 15

(ultimo colore permesso) viene messa a 0, altrimenti viene incrementata.

Se il tasto premuto è ESC (codice 27), viene rimesso lo schermo in modo testo 80 x 25 (`mov Vmode, 3... call VideoMode`) e quindi si ritorna al DOS (int 21h, funz. 4Ch).

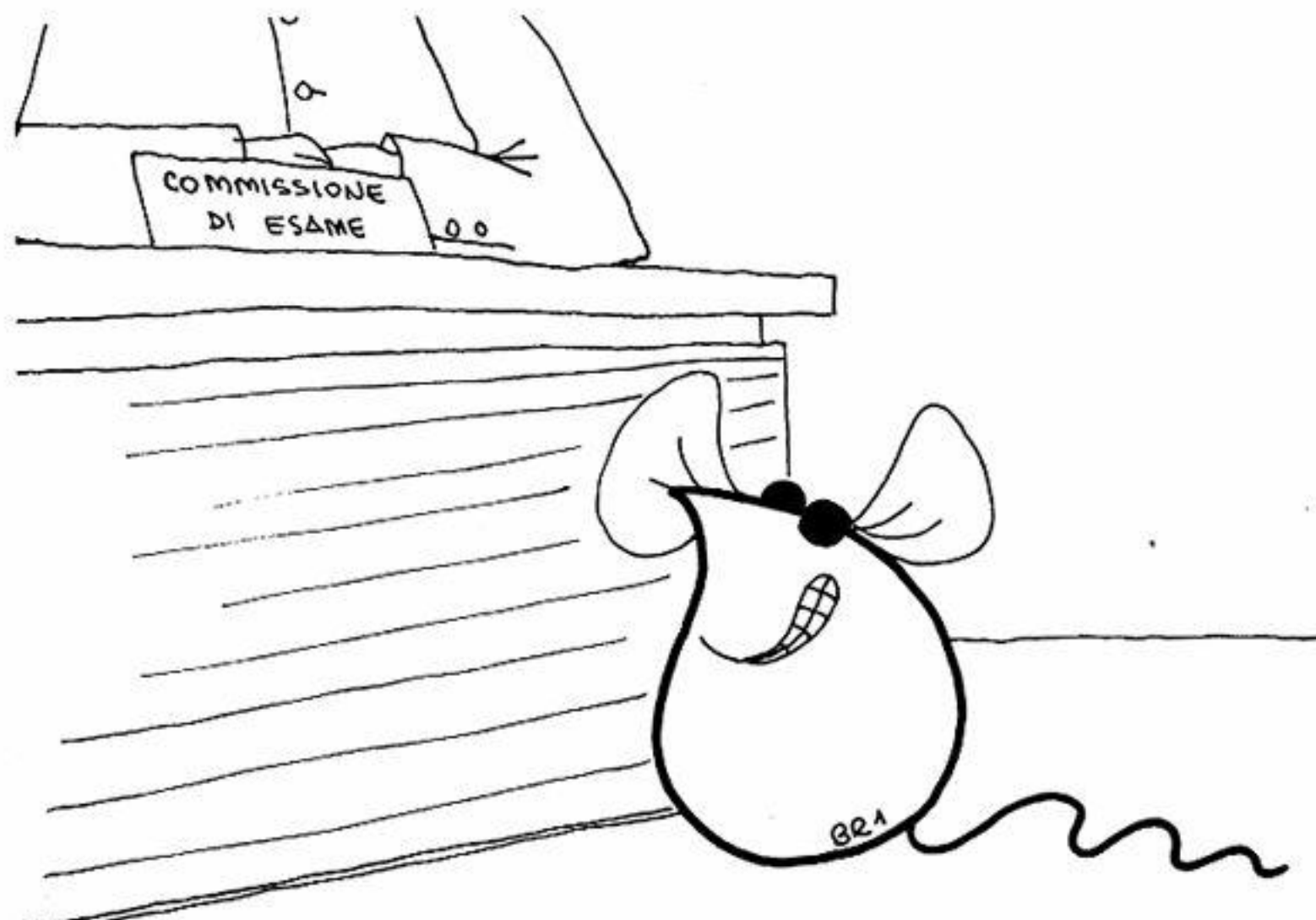
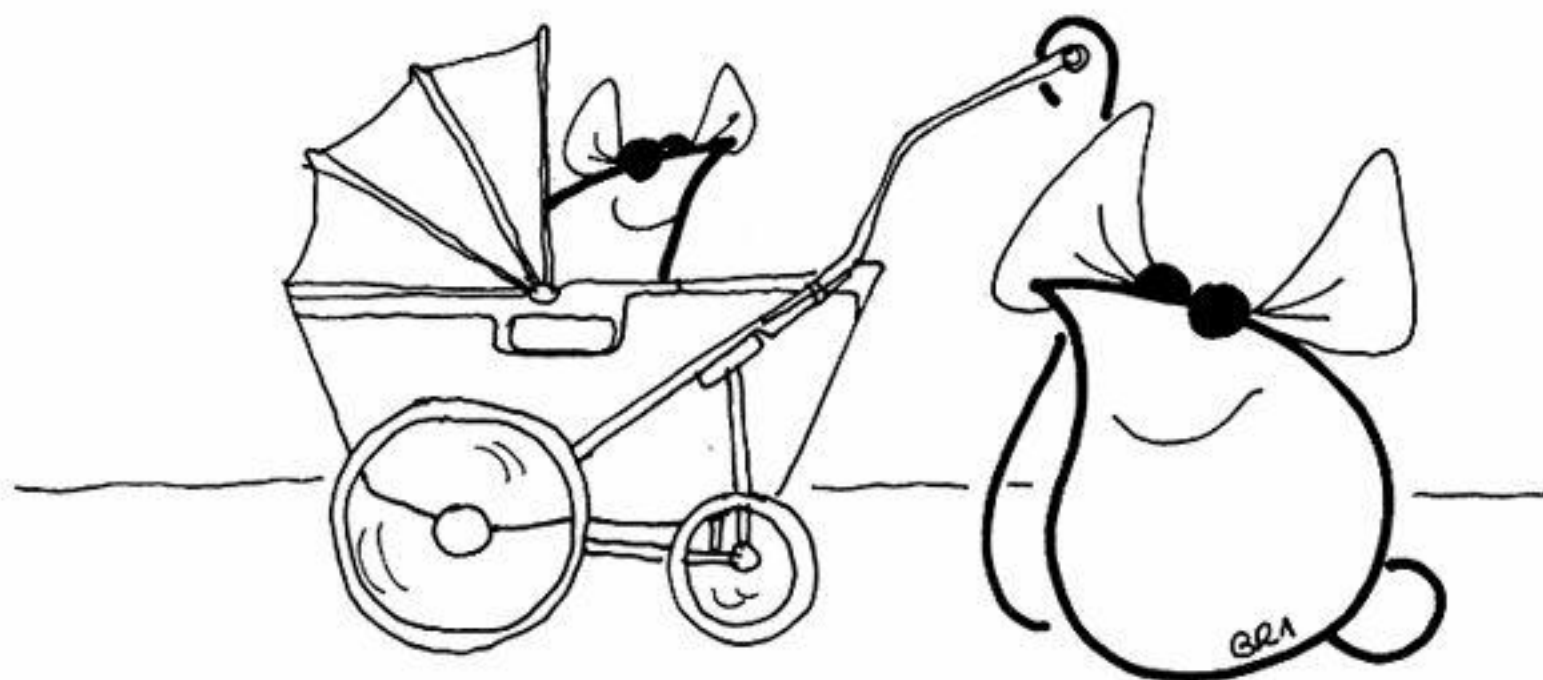
NB: Dal momento che i controlli sui tasti vengono fatti con lettere maiuscole, il tasto Caps Lock deve essere premuto prima di mandare in esecuzione il programma, altrimenti i tasti W, A, X, D, C non verranno riconosciuti.

I limiti delle coordinate non vengono controllati, quindi sarà possibile spostare il cursore fuori dallo schermo. Sarà cura del lettore modificare il programma di conseguenza, inserendo opportune istruzioni CMP... JZ / JNZ nei punti giusti del programma.

Dopo aver digitato il programmino, lo si può registrare con il nome Draw.asm e quindi lo si può compilare tramite le istruzioni:

```
MASM DRAW;
LINK DRAW;
EXE2BIN DRAW.EXE DRAW.COM
DEL DRAW.EXE
```

Richiamando il programma da dos con il comando **Draw** potremo renderci esattamente conto del suo corretto funzionamento. Se tutto è andato bene, vedremo lo schermo cancellarsi e mettersi in modo grafico, e quindi potremo iniziare a schiacciare i vari tasti di movimento e di cambio colore per osservare gli effetti sullo schermo.

Premendo il tasto C da solo si incrementa il colore, quindi in alcuni modi video potrebbe essere non permesso o non visibile. Sarà ancora cura del lettore limitare il cambio di colore al numero di colori permessi dal modo video impostato. **Il programma, così com'è pubblicato, è predisposto per girare su scheda EGA 640 x 350 16 colori.**

di Giancarlo Mariani

Prima ti scrivo, poi ti stampo

Vari modi, in altrettanti linguaggi, di inviare files su stampante o su video

Durante la carriera di un programmatore, sia esso dilettante o professionista, quasi sempre capita di sviluppare programmi che accedono alla stampante, vuoi per realizzare tabulati, per effettuare copie dello schermo, e così via.

In programmi di questo genere sono spesso inserite subroutines diverse per accedere al video oppure alla stampante, anche perchè, normalmente, un foglio di carta può contenere un maggior numero di informazioni rispetto ad una pagina video, e quindi il formato dei dati stampati, o visualizzati, dovrà essere diverso.

Comunque, per molte ragioni, può essere comodo **utilizzare le stesse procedure sia per scrivere su video che su carta**, in modo che un'eventuale modifica al formato di visualizzazione si riproduca immediatamente anche su carta, e viceversa.

Vediamo come sia possibile ciò nei vari linguaggi:

GwBasic e QuickBasic

Come tutti dovrebbero sapere, l'istruzione Basic **Print** consente di scrivere, su video, valori numerici e stringhe alfanumeriche. Simile a questa, esiste la corrispondente istruzione **Print #**, che serve per scrivere su di un file.

Quello che forse non tutti sanno, invece, è che è possibile aprire dei files non solo su disco, ma anche su video e su stampante, consentendo quindi di utilizzare, tra le altre, Print# per scrivere sulle due periferiche.

In Basic, l'istruzione...

```
OPEN "scrn:" FOR OUTPUT AS #xx
```

...ove **xx** è il numero del file, apre il file numero **xx** (compreso tra 1 e 15) sullo schermo, quindi qualsiasi istruzione `Print #xx` effettuata dopo `Open` produrrà una visualizzazione sullo schermo.

Analogamente, l'istruzione...

```
OPEN "lpt1:" FOR OUTPUT AS #xx
```

...apre il file numero xx sulla stampante, quindi qualsiasi istruzione **Print #xx** effettuata dopo Open produrrà un'uscita su carta.

Quindi, inserendo opportunamente nel programma una domanda del tipo *"Vuoi stampare o visualizzare"* e settando il parametro di Open a **scrn:** oppure **lpt1:** sarà possibile utilizzare la stessa procedura per scrivere su video o su stampante.

Un esempio di questa applicazione è dato nel listato **LPrint.bas**, riportato in queste pagine.

Il listato è semplicissimo: tramite l'Input di riga 40, viene richiesto se si vuole

```
{LPRINT.PAS - Borland's Turbo-Pascal 5.5}
{Prova di scrittura su video e stampante}
{ Controlla se la stampante e' collegata}

uses crt,dos;

   var
      stream:text;
     a:char;
     b:string[4];
     regs:Registers;
     v:integer;
begin
  clrscr;
  write ('Video (V) o Stampante (S)? ');
  readln(a);
  b:='';
  if (a='S') or (a='s') then
   begin
      regs.AH:=1;
    regs.AL:=0;
    regs.DX:=0;
    intr(23,regs); { 0x17 }
    v:=regs.AH;
   if ((v and 8) <>0) then
   writeln('La stampante e' spenta o non e'
collegata')
    else
      b:='prn';
  end;
   assign (stream,b);
   rewrite (stream);
  writeln(stream,'Pippo e' andato al mare
a pescare.');
   close(stream);
end.
```

l'uscita su video o su stampante; la riga 60 decide il valore della variabile **B$** a seconda della risposta data.

Le righe da 100 a 120 aprono il file sulla periferica selezionata (video o stampante), quindi scrivono una stringa di esempio e chiudono il file.

Pascal (Turbo, Quick)

Le istruzioni Pascal **Write** e **Writeln** consentono, al pari di Print del Basic, la scrittura di testi e dati, sia su schermo che su file.

Anche nel caso del Pascal, come nel Basic, se vogliamo utilizzare le stesse procedure per scrivere su schermo e su stampante, dovremo aprire un file ed utilizzare le due istruzioni per scrivere sul file.

In Pascal, la coppia di istruzioni...

```
assign (stream,'');
rewrite (stream);
```

...apre il file stream (che deve essere definito come text) sullo schermo (standard output), quindi qualsiasi istruzione `write (stream,...)` oppure `writeln (stream,...)` effettuata dopo le due istruzioni viste prima produrrà un'uscita sullo schermo.

Le istruzioni...

```
assign (stream,'prn');
rewrite (stream);
```

...aprono il file **stream** (che deve essere definito come **text**) sulla stampante, quindi qualsiasi istruzione `write (stream,...)` oppure `writeln (stream,...)` effettuata dopo le due istruzioni viste prima produrrà un'uscita sulla stampante.

Il programma in pascal **LPrint.pas** funziona più o meno come quello in Basic, facendo la domanda *"Video o Stampante?"* e quindi assegnando la variabile **b** alla stringa nulla ("") oppure a **"prn"** a seconda della risposta.

C (Turbo, Quick)

L'istruzione C **printf** consente, al pari di Print del Basic, la scrittura di testi e dati su schermo, mentre la corrispondente **fprintf**, su file.

In questo caso, per utilizzare le stesse procedure sia per la visualizzazione che per la stampa, andrà utilizzata l'istruzione *fprintf* che, con parametri opportuni, consente di scrivere su video o su stampante. Nel caso del C **non** è però necessario aprire dei files, dato che il video e la stampante sono considerati stream sempre aperti.

In C, l'istruzione...

```
fprintf (stdout,..........);
```

...produrrà l'uscita su standard output, quindi su schermo.

Invece, l'istruzione...

```
fprintf (stdprn,..........);
```

...produrrà un'uscita su stampante.

In generale, l'istruzione è...

```
fprintf (stream,..........);
```

...dove **stream** é il file, che può assumere appunto diversi valori. Il programma di esempio **LPrint.C** è un esempio di quanto detto. Il suo funzionamento è simile ai precedenti: a seconda della risposta data alla domanda *"Video o Stampante"* assegna il valore **stdout** oppure **stdprn** alla variabile **stream**, e quindi con `fprintf (stream,....)` scrive una stringa sulla periferica desiderata.

Se la stampante non c'è

Volendo scrivere dati sulla stampante è necessario che questa... sia collegata al computer, accesa e con la carta inserita, altrimenti le istruzioni...

```
Print #
write / writeln (stream,....);
fprintf (stream,...);
```

...producono errore di I/O ed il programma termina immediatamente.

Esiste però un sistema per controllare la presenza di una stampante attiva; consiste nell'usare l'**interrupt 17h**, richiamandolo con il registro **AH = 1** e con il registro **DX = 0**. In questo modo si attiva una routine presente nel Bios del PC che inizializza la porta parallela e ne controlla lo status. La routine restituisce nel registro **AL** del processore i seguenti flags:

bit	Significato
0 =	time out
1 =	N.U.
2 =	N.U.
3 =	i/o error.
4 =	Stampante selezionata
5 =	Fine della carta
6 =	Acknowledge
7 =	Stampante non occupata

```
DEFINT A-Z
'lprintqb.bas
'Prova di scrittura su video e su
'stampante con controllo stamp. presente.
'Solo QuickBasic
'
'richiamare qb con "qb /lqb"

'$INCLUDE: 'd:\qb4\qb.bi'

DIM InRegs AS RegTypeX
DIM OutRegs AS RegTypeX
DIM Outp AS LONG

CLS
INPUT "Video (V) o Stampante (S)"; a$
b$ = "scrn:":
IF UCASE$(a$) = "S" THEN
   InRegs.AX = &H100
   InRegs.DX = 0
   CALL InterruptX(&H17, InRegs, OutRegs)
   a$ = LEFT$(HEX$(OutRegs.AX), 2)
   a = VAL("&h" + a$)
   IF (a AND 8) <> 0 THEN
      PRINT "La stampante e' spenta o non e'
collegata"
   ELSE
      b$ = "lpt1:"
   END IF
END IF

OPEN b$ FOR OUTPUT AS #1
PRINT #1, "Oggi Pippo e' andato al mare"
CLOSE #1
```

Se la stampante non è presente, oppure è in off-line verrà settato il bit **3** del registro **AH** (i/o error) consentendo così di evitare l'errore. Ma come fare a richiamare la funzione nei vari linguaggi?

QuickBasic

In QuickBasic esiste una libreria standard (fornita assieme al programma) chiamata **qb.qlb** che contiene varie istruzioni dedicate al richiamo degli interrupt.

Per utilizzare la libreria, il QuickBasic va richiamato con il parametro "/l" e quindi va specificato il nome della libreria. Un comando per richiamare il qb, quindi, potrebbe essere...

```
qb /l qb.qlb
```

...che richiama il linguaggio utilizzando la libreria **qb.qlb**.

Nel programma che utilizza la libreria deve essere inoltre inserita la riga:

```
'INCLUDE: 'qb.bi'
```

...che, appunto, include nel file Basic sorgente alcune definizioni sulle nuove istruzioni aggiunte.

L'istruzione che ora interessa, tra quelle aggiunte, è la **Interruptx** che consente di richiamare un interrupt da Basic, specificando i registri di ingresso e quelli in uscita.

I registri devono essere contenuti in variabili dichiarate di tipo **RegTypeX**; un programma in QuickBasic che, pertanto, voglia richiamare un interrupt, deve contenere le seguenti istruzioni:

```
'$INCLUDE: 'qb.bi'
DIM InRegs as RegTypeX
'Registri in ingresso
DIM OutResg as RegTypeX
'Registri in uscita
..............
..............
..............
InRegs.AX = xxxx   'Definizione
                    parametri in
InRegs.DX = yyy    'ingresso
InRegs.CX = xxx
CALL InterruptX (Interrupt,
InRegs, OutRegs)
```

I parametri in uscita verranno memorizzati nella variabile **OutRegs**.

Tramite **InRegs** e **OutRegs** si possono leggere e settare tutti i registri del processore 80X86 tranne **CS** e **SS**. Le forme di scrittura saranno, ad esempio...

```
InRegs.AX= xxxx  xxxx= OutRegs.AX
InRegs.BX= xxxx  xxxx= OutRegs.BX
```

...nel caso dei registri AX e BX in ingresso ed in uscita (rispettivamente). Considerazioni del tutto simili varranno per gli altri registri disponibili (CX, DX, BP, SI, DI, FLAGS, DS, ES) identificabili con due o un solo byte.

Per svolgere la funzione vista prima (controllo dello status della stampante) bisognerà quindi scrivere un programmino simile al seguente:

```
'$INCLUDE: 'qb.bi'
DIM InRegs AS RegTypeX
DIM OutRegs AS RegTypeX
................
................
AH=1
'Legge lo status della
'porta parallela dopo averla
'inizializzata.
InRegs.AX = &h0100
InRegs.DX = 0
'Chiama l'interrupt 17h
CALL
InterruptX(&h17,InRegs,OutRegs)
'Legge il valore in ritorno (AH)
Valore = OutRegs.AX
'Sposta il valore
'nella parte bassa
Valore$ = LEFT$(HEX$ (Valore, 2))
Valore = VAL("&h" + Valore$)
'Controlla il bit 3 (i/o error)
IF (Valore AND 8) <> 0 THEN
Print "Stampante non collegata"
```

Si può, inoltre, studiare il programmino di esempio (**LPrintQB.BAS**) per vedere come funziona l'intera procedura.

Pascal (Turbo / Quick)

Anche in Pascal esiste una libreria standard, chiamata **Dos**, che contiene le istruzioni per accedere agli interrupt.

L'istruzione è **intr**, mentre i registri in ingresso ed in uscita devono essere contenuti in una variabile dichiarata di tipo **Register**.

Un programma in Pascal che voglia richiamare un interrupt deve contenere le seguenti istruzioni:

```
uses dos;
var Regs:Register;
...........
...........
Regs.AX := xxxx;   { Definizione
parametri in }
Regs.DX := yyy; {ingresso}
Regs.CX := xxx;
Intr (Interrupt, Regs);
```

I parametri in uscita sono ancora presenti nella variabile **Regs**.

Tramite Regs si possono leggere e settare tutti i registri del processore 80X86, anche nelle forme ad 8 bit. Le forme di scrittura, quindi, saranno analoghe a quelle viste prima (`Regs.AX := xxxx    xxxx := Regs.AX; Regs.AH := xx    xx := Regs.AH`; e così via).

Per svolgere la funzione vista prima (controllo dello status della stampante) bisognerà quindi scrivere un programmino del genere:

```
uses dos;
var Regs:Register;
    Valore:integer;
................
................
{AH=1 : Legge lo status della
porta parallela dopo averla
inizializzata}
Regs.AH := 1;
Regs.AL := 0;
Regs.DX := 0;
{ Chiama l'interrupt 17h
(decimale 23) }
intr (23, Regs);
```

```
10 REM lprint.bas
20 REM Prova di scrittura su video e su stampante.
25 GWBASIC/QUICKBASIC
40 CLS
50 INPUT "Video (V) o Stampante (S)"; a$
60 b$ = "scrn:": IF a$ = "S" THEN b$ = "lpt1:"
100 OPEN b$ FOR OUTPUT AS #1
110 PRINT #1, "Oggi Pippo e' andato al mare"
120 CLOSE #1
```

```
{ Legge valore di ritorno (AH)}
Valore := Regs.AH;
{Controlla il bit 3 (i/o error)}
IF ((Valore AND 8) <> 0) THEN
  WRITELN ('Stampante non
collegata');
```

Si può controllare il programmino di esempio **LPrint.PAS** per vedere come funziona il tutto.

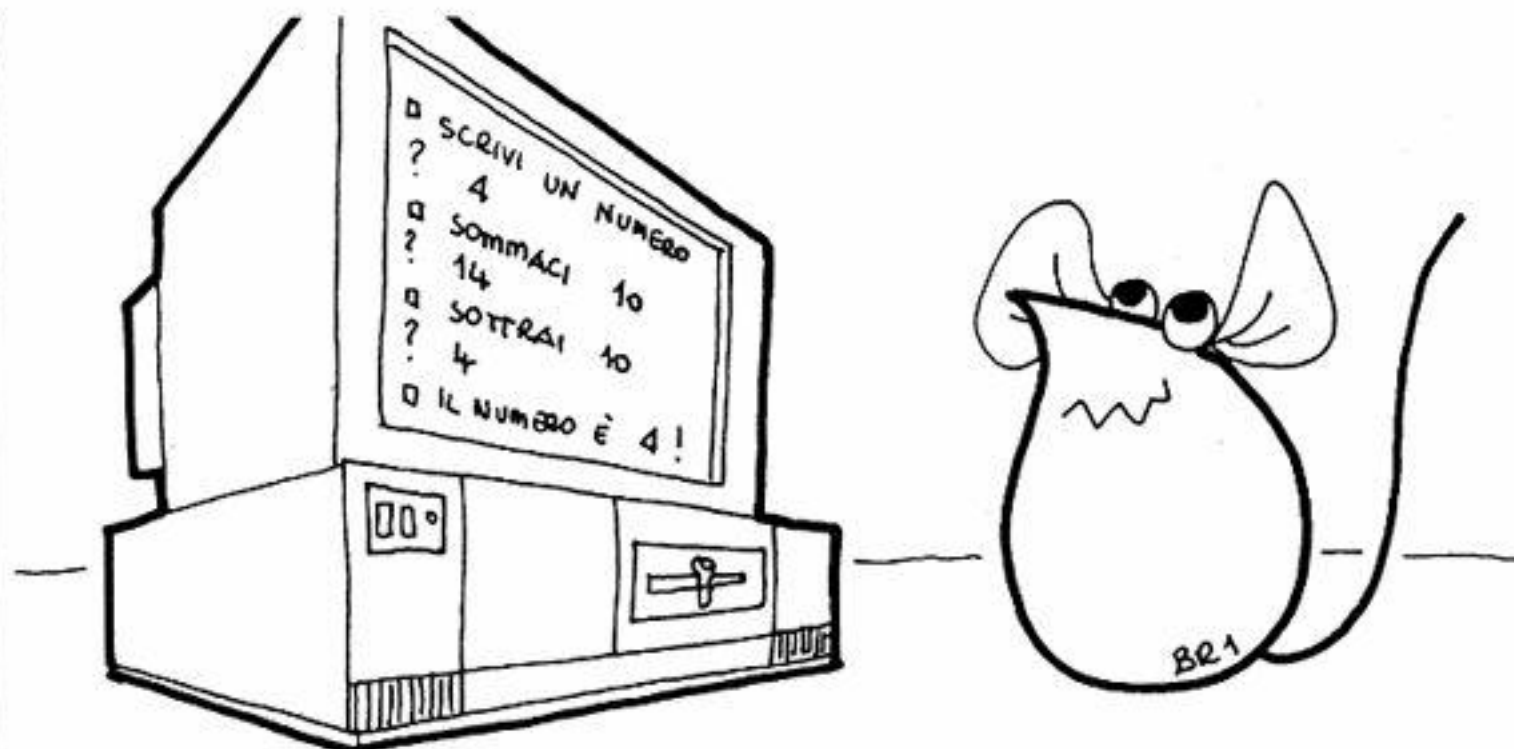
In C

Ovviamente anche in C non poteva mancare la libreria contenente le istruzioni per gestire l'interrupt.

La libreria si chiama ancora **Dos** e, tra le altre, contiene un'istruzione chiamata **geninterrupt** che provvede a richiamare l'interrupt desiderato.

In Turbo-C non è necessario definire una variabile per contenere i registri di ingresso e uscita, dal momento che basta specificare il nome del registro preceduto dal carattere underscore (_, cioè il trattino basso).

Un programma in Turbo-C che voglia richiamare un interrupt deve contenere le seguenti istruzioni:

```
#include <dos.h>
 ...........
 ...........
_AX = xxxx; /*Definizione
parametri in ingresso*/
_DX = yyy;
_CX = xxx;
geninterrupt (Interrupt);
```

I parametri in uscita sono contenuti nelle variabili chiamate come i registri e precedute dal carattere *underscore*.

Tramite le variabili registro si possono leggere e settare tutti i registri del processore 80X86, anche nelle forme ad 8 bit. Le forme di scrittura quindi saranno ancora simili a quelle già viste in precedenza; esempio: _AX = xxxx   xxxx = _AX; _AH = xx   xx = _AH; e così via.

Per svolgere la funzione vista prima (controllo dello status della stampante) bisognerà quindi scrivere un programmino del genere:

```
#include <dos.h>
unsigned int Valore;
................
................
/* AH=1 : Legge lo status della
porta parallela dopo averla
inizializzata */
_AH = 1;
_AL = 0;
_DX = 0;
/* Chiama l'interrupt 17h */
geninterrupt (0x17);
/* Legge il valore in ritorno
(AH) */
Valore = _AH;
/*Controlla il bit 3 (i/oerror)*/
if ((Valore &&8)!= 0)
     printf("Stampante non
collegata\n");
```

Si può controllare il programmino di esempio **LPrint.C** per vedere come funziona il tutto.

```
/* Lprint.c - Borland's Turbo-C 2.0
*/
/* Permette di scrivere su video o su
stampante*/
/*Controlla se la stampante e' presente*/

#include <stdio.h>
#include <conio.h>
#include <dos.h>

void main()
{
   FILE *stream;
   char a;
  unsigned int v;

   clrscr();
   printf("Video (V) o Stampante (S)? ");
   scanf("%c",&a);
   stream=stdout;
   if (a=='S' || a=='s')
   {
      _AH=1;
     _AL=0;
     _DX=0;
     geninterrupt(0x17);
     v=_AH;
    if ((v && 8) !=0)
    printf("La stampante e' spenta o non e'
collegata\n");
     else
      stream=stdprn;
   }
   fprintf(stream,"Pippo e' andato al mare a
pescare\n");
}
```

di Alessandro Marrazzo

3 mini file batch

La stesura di semplici file batch sono di aiuto per i principianti

E' noto che i file **Ascii** contenenti i comandi del Dos rappresentano veri e propri programmi, attivabili direttamente digitandone il nome, pur se viene omesso il suffisso **.BAT**.

Invitiamo i lettori, specie se principinati, a documentarsi in merito esaminando il manuale di istruzioni del proprio computer Ms - Dos compatibile.

D.BAT

Sintassi: D [drive:] [pathname]

```
@ ECHO Digita: D [drive:] [PATH]
@ ECHO Directory completa
@ ECHO su path "%1"
@ DIR %1 /P
@ EXIT
```

N.B: registrare questo file con il nome D.BAT

Il "comando" **D.BAT** serve a visualizzare una directory tramite il comando **Dir**, usato nella sua sintassi più frequente e cioè con lo switch **/P** che interrompe il listing alla fine di ogni pagina video.

Questo primo file, di estrema semplicità, serve per vedere alcune cose interessanti. La prima è la "variabile" **%1** che, nel caso di un file di tipo batch, conterrà il gruppo di caratteri digitati dopo il nome del file stesso (nel nostro caso il Path, eventualmente preceduto dal drive); la seconda è il carattere di chiocciola (@) presente all'inizio di ogni riga del file. Questo carattere rappresenta una maniera più elegante (e nella programmazione la cosa non guasta) per impedire l'emissione a video dei comandi che a mano a mano vengono eseguiti, evitando così anche la fastidiosa visualizzazione del comando **Echo Off**.

Un altro sistema per ottenere lo stesso risultato, è quello di usare, come sempre, Echo Off, preceduto però dal carattere chiocciola, che in questo modo non avrà più ragione di essere usato, fino ad un nuovo comando Echo On. L'ultimo comando è Exit che, come l'End del Basic, ha la funzione di chiudere il programma. Per esercizio, consigliamo ai principianti di modificare il file proposto in modo da utilizzare anche l'opzione **/W**, che permette di visualizzare solo i nomi dei files (privi della descrizione) fino a 5 per ogni riga video.

E.BAT

Sintassi: E estens. [drive:] [pathname]

```
@ ECHO E estens. [drive:] [path]
@ ECHO Directory dei files
@ ECHO con est. "%1" su path "%2"
@ IF "%2" == "" GOTO NOPATH
@ DIR %2\.%1 /P
@ GOTO EXIT
: NOPATH
@ DIR .%1 /P
: EXIT
@ EXIT
```

N.B: assegnare a questo file il nome E.BAT

Con il file **E.BAT** possiamo visualizzare, sempre tramite il comando interno Dir, la directory di tutti i file aventi una particolare **estensione**, eventualmente specificando anche il Path.

Esaminando il primo file (D.BAT) abbiamo visto che digitando un qualsiasi gruppo di caratteri dopo il nome di un file batch, questi possono essere richiamati, durante l'esecuzione, con la variabile **%1**; adesso possiamo notare la presenza di un'altra variabile (**%2**) che conterrà il secondo gruppo di caratteri, eventualmente digitati dopo il primo, e separati da questo con uno **spazio**.

Nel nostro caso viene offerta la possibilità di specificare un drive e/o un path del quale visualizzare la directory selezionata. La terza riga del programma, tramite l'istruzione **IF** ed il simbolo di assegnazione uguale uguale (**==**) permette di verificare se è stato digitato anche un secondo parametro dopo il primo: in caso negativo, avvalendosi dell'istruzione **GOTO**, si effettua un salto all'etichetta specificata (NOPath); altrimenti, proseguendo normalmente, viene eseguito Dir assegnando a **%2** il path specificato. Saltando all'etichetta Exit il programma termina.

LIST.BAT

Sintassi: LIST filename.ext

```
@ ECHO SINTASSI: LIST filename.ext
@ ECHO File: %1
@ TYPE %1 | MORE
@ EXIT
```

N.B: assegnare a questo file il nome LIST.BAT

Abbiamo visto che quando dobbiamo visualizzare directory molto lunghe abbiamo a disposizione l'opzione **/P** che permette di bloccare lo scorrimento su video in modo da non perdere informazioni. Per ottenere lo stesso scopo, il Dos mette a disposizione anche un altro sistema (il comando **More**) che, operando come un *filtro*, legge l'input che gli viene indirizzato, crea su disco un file temporaneo nel quale viene memorizzato il contenuto, e poi lo visualizza una videata per volta (ne consegue che se il disco è protetto da scrittura, l'operazione non va a buon fine). Una frequente applicazione della procedura si verifica quando, con il comando Type, vogliamo esaminare a video il contenuto di un file Ascii. Il comando List.bat permette appunto di effettuare l'operazione associando a Type il comando More per ottenere l'interruzione temporanea dell'emissione a video, senza doverlo digitare ogni volta.

di Graziano Settimini

Se la calcolatrice non basta più

La matematica sembra avere molti sostenitori; ecco lo studio, decisamente interessante, di un nostro "Amigo"

L'articolo di queste pagine, riguardante la matematica, è la logica continuazione del programma apparso sul numero 82, che riportava un listato atto al processo di "divisione infinita", ossia in grado di individuare un quoziente con un numero n di cifre decimali.

Per completarlo *(a parte il suggerimento di esaminare il lavoro di altri due lettori che hanno risposto esplicitamente alla sfida, ndr)* si riporta il metodo per evitare alcuni sottili errori che si verificavano con tale programma, in particolare per quanto riguardava il rilevamento delle cifre costituenti il periodo (basti menzionare il caso di...

123 / 4567 = 0.02693234070505

...in cui il computer forniva come periodo il gruppo "05").

Poiché un periodo di due cifre è generato solo da divisori multipli di **11**, si rivelerebbe necessario controllare tale evenienza.

Analogamente, i periodi di tre cifre vengono generati solo dai divisori multipli di **37**, per cui è necessario un altro **If ... Then**; "falsi periodi" costituiti da un numero maggiore di cifre compaiono molto raramente e solo con divisori molto grandi, quindi si possono tranquillamente tralasciare.

La modifica che si può suggerire (a coloro che hanno già digitato il listato del n. 82 e non hanno voglia di scrivere i pur semplicissimi programmi riportati su altra parte della rivista) riguarda le due righe seguenti:

```
485 if x$=y$ and i>1 and len(x$)=2
    and d1/11<>int(d1/11) then 530
486 if x$=y$ and i>1 and len(x$)=3
    and d1/37<>int(d1/37) then 530
```

...che, pur non risolvendo completamente il problema, rappresentano un valido surrogato.

Moltiplicando

In queste pagine è presentato un programma di moltiplicazione (quasi) infinita, che serve, oltre che a giustificare il titolo dell'articolo, a **determinare il prodotto di due numeri rappresentati da qualche centinaio di cifre decimali**, ottenuti (magari) con il programma di divisione (quello comparso sul pluricitato n. 82, per intenderci), come richiesto dalla sfida ivi lanciata.

Grazie al formato di sottoprogramma, cioè totalmente trasportabile, sarà semplice calcolare moltiplicazioni di numeri interi molto grandi anche in vostri programmi.

Esaminiamo ora i principi logici che sono alla base del programma.

L'algoritmo

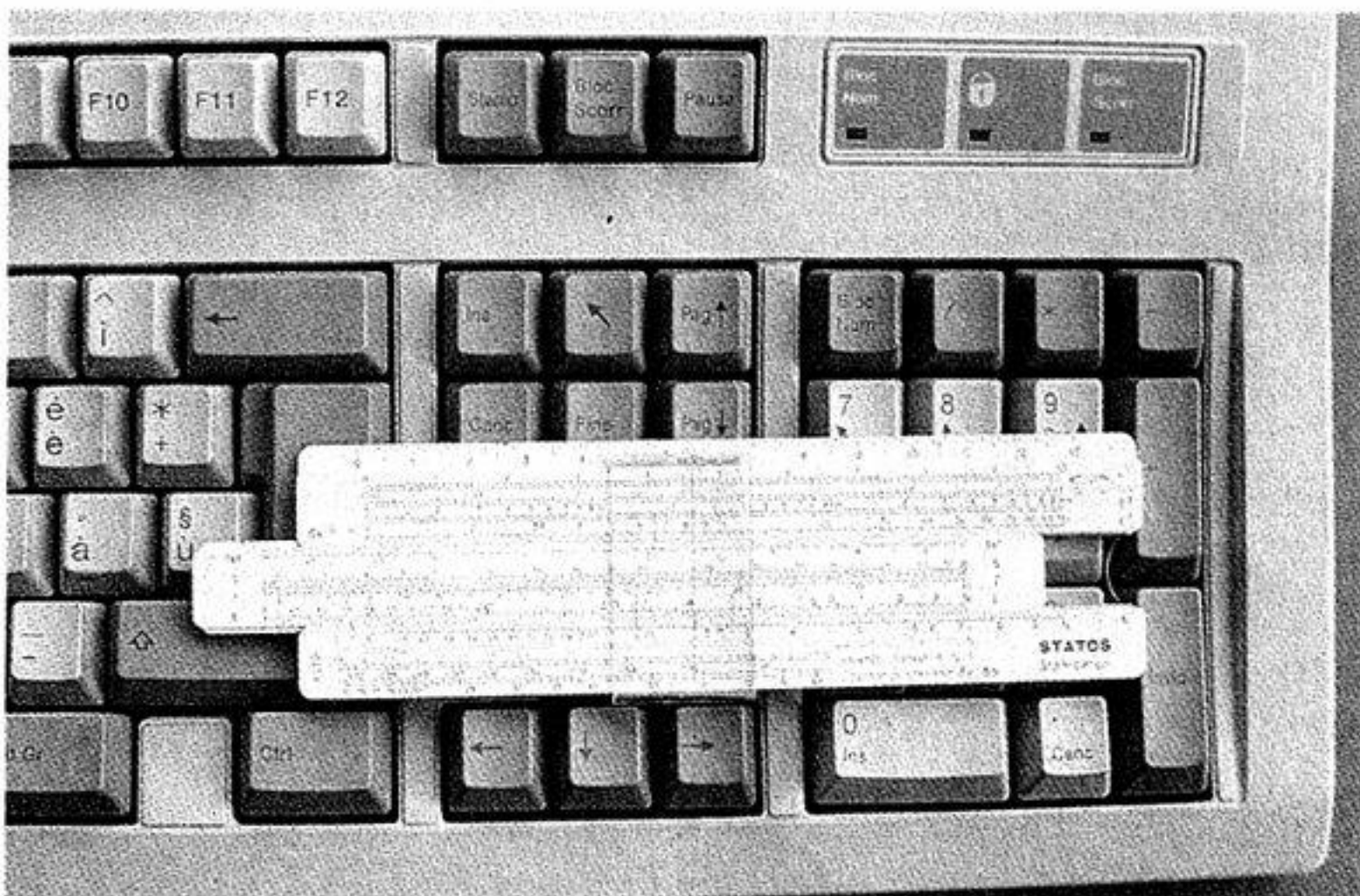
Poiché i numeri in gioco sono composti da moltissime cifre, tali da non poter essere direttamente gestibili dal computer, è d'obbligo operare con le stringhe, e far eseguire al nostro **Amiga** le somme e le moltiplicazioni proprio come faremmo noi (umani) con carta e penna.

I passi da compiere (algoritmo) sono i seguenti:

1) si prende una cifra del secondo fattore (da destra verso sinistra, ordinatamente);

2) la si moltiplica per il primo fattore, cifra per cifra;

3) il prodotto ottenuto è un risultato parziale, da moltiplicare per quella poten-

za di 10 ottenuta sottraendo 1 dalla n-esima cifra della fase uno;

4) finché non si raggiunge l'ultima cifra, si salta alla fase uno;

5) si esegue la sommatoria dei risultati parziali.

Più facile a farsi che a dirsi, dirà qualcuno, ma l'attento lettore avrà notato che la fase *cinque* ha bisogno di una ulteriore definizione attraverso un algoritmo, in quanto essa non fa altro che svolgere delle somme fra numeri costituiti da varie centinaia di cifre; funzione, questa, non direttamente traducibile in Basic:

a) si somma una cifra del primo addendo (da destra verso sinistra, ordinatamente) con la cifra corrispondente del secondo addendo e con il riporto precedente;

b) si ottiene un numero di DUE cifre, di cui la prima (decine) è il riporto, mentre la seconda fa parte del totale;

c) finché non si raggiunge l'ultima cifra, si salta alla fase a.

Questi step si ritrovano pari pari (o quasi) nei sottoprogrammi dai nomi fantasiosi di **Prodotto** e **Somma**.

Una precisazione sull'algoritmo di addizione: i due addendi devono tassativamente essere interi, per garantirne il perfetto **incolonnamento** (condizione indispensabile per un risultato corretto), il che è assicurato dalla fase tre. Gli algoritmi presentati valgono unicamente per la somma ed il prodotto di DUE numeri.

Non resta infine che l'**elevamento a potenza**: poiché tale operazione è formalmente denotata dal simbolo a^b (dove **a** è la base e **b** è l'esponente), ed è definita come il prodotto **a** x **a** x **a** x ... b volte; esempio:

13^5 = 13 x 13 x 13 x 13 x 13 (5 volte)
7^{11} = 7 x 7 x 7 x 7 x 7 x 7 x 7 x 7 x 7 x 7 x 7 (11 volte)

Sarà dunque sufficiente ripetere **b** volte l'algoritmo di moltiplicazione tra due fattori.

Il programma

Dal momento che il programma di moltiplicazione accetta in Input (anche) i dati provenienti dal programma di divisione (immaginate se dovevate digitarli?), dovremo modificare quest'ultimo in modo che "scarichi" tali dati in un luogo da cui poi recuperarli: è stata scelta, come device di output, la RAM DISK, ma, a scelta, si può utilizzare anche un dischetto semplicemente sostituendo **RAM:** con **DF0:** (oppure con **HD1:** per l'hard disk) senza trascurare di specificare il percorso completo ove memorizzare il file, se non lo si desidera nella directory principale (root).

Nel complesso, per rendere la primitiva versione del programma di divisione compatibile col programma di queste pagine, sono necessarie le seguenti modifiche:

- cancellare le righe 240 / 245 / 250 / 255 / 260 / 385 / 386 / 387
- aggiungere le righe:

```
220 OPEN"RAM:fattori"FOR OUTPUT AS #1
230 FOR ciclo = 1 TO 2: CLS: a$ =
    "": i = 0: co = 1
280 IF cd <4 OR cd > 32767 OR cd >
    FRE(1) THEN PRINT "ERRORE":
    GOTO 270
370 i = i + 1: IF i = cd THEN
    PRINT#1, STR$ (dx); a$;
    CHR$ (10); LEN (a$): GOTO 460
520 IF ciclo = 2 THEN 540 ELSE PRINT
    "premi un tasto"
530 IF ciclo = 1 AND INKEY$ = ""
    THEN 530
540 NEXT ciclo: CLOSE #1
```

A questo punto salvate il programma così ottenuto su dischetto.

Come si è detto, in riga 220 la device (RAM:) e il file ("fattori") sono modificabili a piacimento (in tal caso non dimenticate di modificare anche il programma di moltiplicazione nel comando OPEN di apertura del file, verso l'inizio dello stesso).

Con la riga 280 ci si libera dagli angusti limiti delle 254 cifre decimali degli otto bit, sfruttando tutta la potenza dei sedici bit di Amiga, che può gestire **stringhe di 32767 caratteri**; con FRE(1) si controlla che la lunghezza richiesta della stringa-quoziente non ecceda il limite della memoria disponibile; l'assegnazione **Co = 1** è indispensabile per garantire la compatibilità.

La riga 370 adempie al compito di riversare nel file le due stringhe-quozienti, che nella successiva elaborazione saranno trattate come stringhe-fattori, secondo il formato:

Quoziente senza virgola (lf)
Numero di cifre decimali (lf)

I due campi sono separati da un **Line feed = chr$(10)**.

In pratica, il numero **157,09** sarebbe memorizzato così:

15709 (LF)
2 (LF)

Inutile dire che non è assolutamente necessario utilizzare il programma pubblicato sul n. 82 prima di attivare quello che compare in queste pagine: è infatti sufficiente avere a disposizione, su dischetto, un file che contenga (secondo il "codice" appena indicato) i valori richiesti.

Esaminiamo ora il primo sottoprogramma, quello relativo alla moltiplicazione. Esso accetta in input due fattori (memorizzati in altrettante variabili stringa) e restituisce, in output, l'array **prod$()**, che contiene tutti i risultati parziali: all'uscita dal sottoprogramma siamo giunti alla fase cinque dell'algoritmo.

Da notare l'uso della funzione **String$**, utile per moltiplicare il risultato parziale per l'appropriata potenza di 10 (vedi la fase tre), che rende il tutto più agile; resta da dire che il primo ciclo FOR scandisce ogni cifra del secondo fattore, la quale moltiplica ogni cifra del primo grazie al secondo ciclo FOR ... NEXT: le variabili **a** e **b** contengono, rispettivamente, la cifra corrente del primo e del secondo fattore, mentre la variabile **carry** (in inglese: riporto) contiene proprio il riporto derivante da ciascuna moltiplicazione parziale.

Il sottoprogramma dell'addizione, invece, accetta in input due addendi, memorizzati in variabili stringa, e fornisce, strano a dirsi, la loro somma. La prima parte aggiunge in testa alla stringa più corta (= all'addendo col minor numero di cifre) degli zero non significativi, giusto per pareggiare i conti, ed evitare problemi con MID$; per il resto la comprensione è immediata, magari tenendo sott'occhio l'algoritmo, la cui struttura il sottoprogramma rispecchia fedelmente.

A questo punto, per completare l'esecuzione della moltiplicazione, è necessario richiamare la subroutine "somma" per tutti i risultati parziali forniti in output dalla subroutine "prodotto". Lo schema della procedura Basic da seguire per una generica moltiplicazione è dunque:

```
CALL prodotto(...)
FOR x=1 to n
CALL somma(...)
NEXT x
```

L'indice massimo di DIMensionamento dell'array **prod$** è pari al numero delle cifre del secondo fattore.

Condizione vitale per il buon funzionamento del tutto è che entrambi i sottoprogrammi ricevano in input stringhe contenenti solo cifre e NON punti decimali, vir-

I tempi di elaborazione

Per la costituzione dell'algoritmo stesso, il tempo impiegato dal nostro **Amiga** per portare a termine una moltiplicazione è proporzionale al prodotto delle lunghezze, in cifre, dei due fattori...

t = k x l1 x l2

...ove **t** è il tempo necessario ad eseguire l'operazione, **l1** è il numero di cifre del primo fattore, **l2** è il numero di cifre del secondo fattore e **k** è una costante che permette di determinare il tempo di elaborazione espresso in secondi, e che per l'**AmigaBasic** assume il valore di **0.056**: ciò significa che moltiplicando due numeri di **100 cifre ciascuno** sono necessari...

t = 0,056 x 100 x 100 =
= 0,056 x 10000 = 560 secondi

...che equivalgono a circa 9 minuti e 20 secondi.

In particolare, se **l1 = l2**, vale a dire se i fattori hanno lo stesso numero di cifre, si ha che $\mathbf{t = k \times l^2}$, cioè che il tempo aumenta con andamento parabolico, dunque impercettibilmente per bassi valori di **l** ma molto velocemente per valori di **l** medio- alti.

Il tempo richiesto dall'operazione di elevamento a potenza, invece, è dato dalla formula...

$\mathbf{t = k \times (l \times b)^2}$

...ove **t** è il tempo necessario al calcolo della potenza $\mathbf{a^b}$; **b** è appunto l'esponente ed **l** è il numero di cifre costituenti la base **a**, ed infine **k** è ancora la costante temporale, pari a **0,018**. A questo punto siamo in grado di risolvere anche problemi pratici: ad esempio, se vogliamo sapere quanti bytes di memoria può indirizzare un microprocessore il cui bus indirizzi è a 64 bit, è sufficiente(!) calcolare 2^{64}, ma poiché tale idea non sfiora nemmeno la mente dei comuni esseri raziocinanti, ecco che con questo programma verranno spalancate le meravigliose porte della microelettronica! Prima però di dare il via ai calcoli sarebbe bene conoscere a priori quanto tempo sarà impiegato, giusto per sapere se è il caso di spenderlo in modi migliori: ed ecco che salta fuori la necessità della formula del tempo. In questo caso...

$\mathbf{t = 0{,}018 \times (1 \times 64)^2 =}$
0,018 x 4096 = 73,72 secondi,

...equivalenti a circa 1 minuto e 13 secondi.

Non crucciatevi, dicevamo, se il massimo esponente permesso è **32767**, perché per esponenti maggiori rischiereste di non campare abbastanza a lungo per essere presenti alla visualizzazione del risultato (lasciarla in eredità ai posteri?): infatti già per calcolare il valore di 6^{32767}, benché la base sia costituita da una sola cifra, dovrete attendere la bellezza di...

$t = 0{,}018 \times (1 \times 32767)^2 =$
19326173 secondi

...equivalenti a **223 giorni**, 16 ore, 22 minuti e 53 secondi, sufficienti a scoraggiare anche il matematico più puramente teorico, senza contare che, tradotti in soldoni, significano un bel gruzzolo all'Enel!

gole o frattaglie varie: nel programma di queste pagine si assicura ciò mediante la particolare forma di memorizzazione nel file (ricordate?), ma nei vostri programmi dovrete vedervela da soli! Ricordiamo, a tal proposito, che per essere trasportata nei vostri lavori, la subroutine non necessita della minima modifica da parte vostra.

Poiché la moltiplicazione viene realmente effettuata fra numeri interi, il problema di dove vada a finire la virgola si risolve tenendo presente che *"il numero di cifre decimali del prodotto è pari alla somma delle cifre decimali dei due fattori"*; ora, queste ultime sono state memorizzate apposta nel file, per cui mediante la banale somma **v = a + b** si recupera la posizione occupata in partenza dalla virgola, grazie alla funzione stringa definita nella prima riga del programma.

Infine, finalmente, il prodotto verrà visualizzato.

Per quanto riguarda il calcolo delle potenze, è sufficiente inserire in un ciclo For ... Next la procedura vista prima relativa ad una moltiplicazione. Dal punto di vista tecnico c'è da dire che i dati in input provengono soltanto da tastiera; inoltre, mentre la **base**, essendo memorizzata per mezzo di INPUT in una variabile stringa, può essere costituita da un qualsivoglia numero di cifre, l'**esponente** deve essere un numero intero compreso tra **0** e **32767**, poiché viene memorizzato in una variabile dichiarata di tipo "intero breve": cambiando la dichiarazione potrete aumentare il valore massimo accettato per l'esponente, ma anche i tempi di elaborazione!

Il programma non accetta esponenti decimali e neppure negativi, al fine di non complicare esageratamente una routine di elevamento a potenza, che trova la sua unica ragione di esistere solo nella sua brevità e semplicità.

Istruzioni

Dopo tutta questa marea di teoria, è arrivato il momento di vedere come metterla in pratica. Per quanto concerne la divisione, non essendo state apportate modifiche sostanziali, si rimanda all'articolo sul numero 82.

Doveroso, invece, illustrare l'uso del programma, peraltro molto semplice. Una volta digitato il listato, controllato che non vi siano errori di sorta, salvato su un dischetto, e lanciato con RUN, comparirà una domanda, cui dovrete rispondere premendo **M** per eseguire una moltiplicazione, oppure **P** per un elevamento a potenza.

Scegliendo la moltiplicazione (opzione M), non dovrete far altro che aspettare, poiché il computer leggerà il file dei due fattori, li stamperà sullo schermo, e, dopo un certo lasso di tempo (sapete già "quanto"), in cui il vostro Amiga non darà il minimo segno della turbinosa attività che lo anima, comparirà il prodotto cercato.

Se, invece, scegliete l'operazione di elevamento a potenza (opzione P), comparirà immediatamente la richiesta della base, che dovrà consistere in un numero positivo (non specificate il segno + per carità...) senza limitazioni sul numero di cifre, e, subito dopo, quella dell'esponen-

```
DEF FNvirg$ (x$, x)= LEFT$ (x$, LEN (x$) -x)+ "," +RIGHT$ (x$,x)
DEF FNtrunc$ (x$) = RIGHT$ (x$, LEN (x$) -1)
1 CLS: INPUT "Moltiplicazione o Potenza"; a$
IF UCASE$ (a$) = "P" GOTO potenza
IF UCASE$ (a$) <> "M" GOTO 1
moltiplicazione:
OPEN "ram: fattori" FOR INPUT AS #1
INPUT#1, a$, a, b$, b: v = a + b: CLOSE #1
PRINT "Primo fattore :" FNvirg$ (a$, a)
PRINT "Secondo fattore:" FNvirg$ (b$, b)
DIM c$ (LEN (b$) ): x$ = ""
CALL prodotto (a$, b$, c$ () )
FOR i = 1 TO UBOUND (c$)
CALL somma (c$ (i), x$, r$)
x$ = r$: NEXT i
r$ = FNvirg$ (r$, v)
WHILE LEFT$ (r$, 1) = "0": r$ = FNtrunc$ (r$): WEND
PRINT "Risultato = "r$: END
potenza:
INPUT "Base"; a$: l = LEN (a$): v = INSTR (a$,".")
IF v> 0 THEN a$ = LEFT$ (a$, v-1) + MID$ (a$, v + 1)
INPUT "Esponente"; b%: DIM s$ (LEN (a$) ): c$ = a$
IF v> 0 THEN v = (l-v) *b%
FOR i = 1 TO b%-1: x$ = ""
CALL prodotto (c$, a$, s$ () )
FOR s = 1 TO UBOUND (s$)
CALL somma (s$ (s), x$, c$)
x$ = c$: NEXT s
NEXT i
WHILE ASC (c$) = 48: c$ = FNtrunc$ (c$): WEND
IF v> 0 THEN c$ = FNvirg$ (c$, v)
PRINT "risultato = "; c$: END

SUB somma (add1$, add2$, tot$) STATIC
tot$ = "": carry = 0
IF LEN (add1$) > LEN (add2$) THEN
WHILE LEN (add1$) > LEN (add2$): add2$ = "0" + add2$: WEND
ELSE
WHILE LEN (add2$) > LEN (add1$): add1$ = "0" + add1$: WEND
END IF
FOR cifra = LEN (add1$) TO 1 STEP -1
a = VAL (MID$ (add1$, cifra, 1) )
b = VAL (MID$ (add2$, cifra, 1) )
c = a + b + carry: carry = INT (c/10)
tot$ = RIGHT$ (STR$ (c), 1) + tot$
NEXT cifra
END SUB

SUB prodotto (fatt1$, fatt2$, prod$ () ) STATIC
l1 = LEN (fatt1$): l2 = LEN (fatt2$)
FOR i = l2 TO 1 STEP -1: carry = 0
prod$ (i) = STRING$ (l2-i,"0"): b = VAL (MID$ (fatt2$, i, 1) )
FOR k = l1 TO 1 STEP -1
a = VAL (MID$(fatt1$, k, 1)): p = a*b + carry: carry= INT (p/10)
prod$ (i) = RIGHT$ (STR$ (p), 1) + prod$ (i)
NEXT k
prod$ (i) = RIGHT$ (STR$ (carry), 1) + prod$ (i)
NEXT i
END SUB
```

Il listato in AmigaBasic
La brevità del programma invita a sperimentare sofisticazioni di ogni genere

te, che, lo ricordiamo, dovrà essere un numero intero compreso tra 2 e 32767.

Anche in questo caso le vostre fatiche terminano qui, in quanto dovrete solo aspettare (od occupare meglio il vostro tempo) che l'Amiga vi sforni il risultato.

Ancòra, la formula del tempo sarà elogiata per non avervi fatto passare notti insonni prima che comparisse il risultato tanto agognato.

Concludendo

Un caloroso consiglio: se potete, compilate il programma, in modo da abbassare la costante K e rendere tutte le operazioni più spedite; la presenza del coprocessore matematico 68881 **non** dovrebbe influenzare il ritmo delle operazioni (per il motivo che non vengono eseguiti, internamente, calcoli in virgola mobile) lasciando sostanzialmente inalterato il valore di k.

A proposito del programma, in esso manca quasi ogni tipo di controllo: sarebbe opportuno che il lettore inserisse dei "filtri" software che impediscano l'input di dati non ammessi, nonché un segmento di programma che renda inoffensivi eventuali errori che potrebbero verificarsi in fase di esecuzione del programma: a parte i banali Syntax error, dovrebbero essere "parati" in particolare il File not found (il file "fattori" deve effettivamente esistere nella device selezionata) e l'Out of memory (utilizzare l'istruzione **Clear** per allocare una maggior quantità di memoria). Negli Amiga 512K sarà opportuno non utilizzare più di 200 K di RAM, per mezzo dell'istruzione `CLEAR, 25000: CLEAR, 200000`; se possedete un'espansione di memoria potrete sbizzarrirvi liberamente.

Altra importante e sostanziale modifica sarebbe l'implementazione di algoritmi più efficienti e quindi più veloci di quelli proposti. Datevi quindi da fare e utilizzate questo programma anche come palestra software, nonché come ennesima sfida, che qui sintetizziamo:

Scrivere un programma che riduca drasticamente il tempo di elaborazione richiesto dal listato pubblicato in queste pagine.

Credete che sia troppo difficile? Bè, non penserete che la Systems Editoriale regali facilmente mille lire al primo aspirante-collaboratore che capita in Redazione...

di Andrea Preziosi

Se premi un tasto parte un programma

Come personalizzare le funzioni svolte dai tasti del nostro PC

Qualsiasi personal computer Ms - Dos, non appena viene acceso, esegue una procedura nota con il nome di autodiagnostica, grazie alla quale, in pratica, il computer prende "visione" dell'hardware installato e controlla se tutto funziona bene.

In caso affermativo viene eseguito il **Bootstrap**, termine che letteralmente vuol dire *"con i lacci degli stivali"*, frase molto strana che, tuttavia, non è fuori luogo come potreste pensare: un uomo che si alza aggrappandosi ai lacci dei suoi stivali (praticamente impossibile) non fa ricorso ad appoggi esterni; allo stesso modo un personal computer, per iniziare a funzionare, esegue i programmi contenuti nelle sue **Rom**, senza far ricorso a mezzi esterni.

Il computer, quindi, cerca sul drive **a:** un programma contenuto nel settore di **Boot** (nel seguito denominato *loader*, cioè caricatore); se il floppy non è formattato, oppure il disk-drive non c'è o risulta privo di dischetto, il computer carica dall'hard-disk il loader, programma in grado di far partire il computer. La procedura, tra le altre cose, carica ed attiva due file "nascosti", termine con il quale si intendono dei file non gestibili direttamente dal Dos.

In effetti è possibile "nascondere" qualsiasi file, semplicemente modificando i suoi attributi con programmi di utilità evoluti (tipo **PcTools**).

Nei floppy privi di boot, il loader è sempre presente, ma inefficace, nel senso che si limita a visualizzare un messaggio in cui si invita l'utente a sostituire il floppy con uno che contiene i file vitali del Dos. Ricordiamo (vedi numeri precedenti della nostra rivista), che è possibile modificare il messaggio visualizzato, purché se ne rispetti la lunghezza originale.

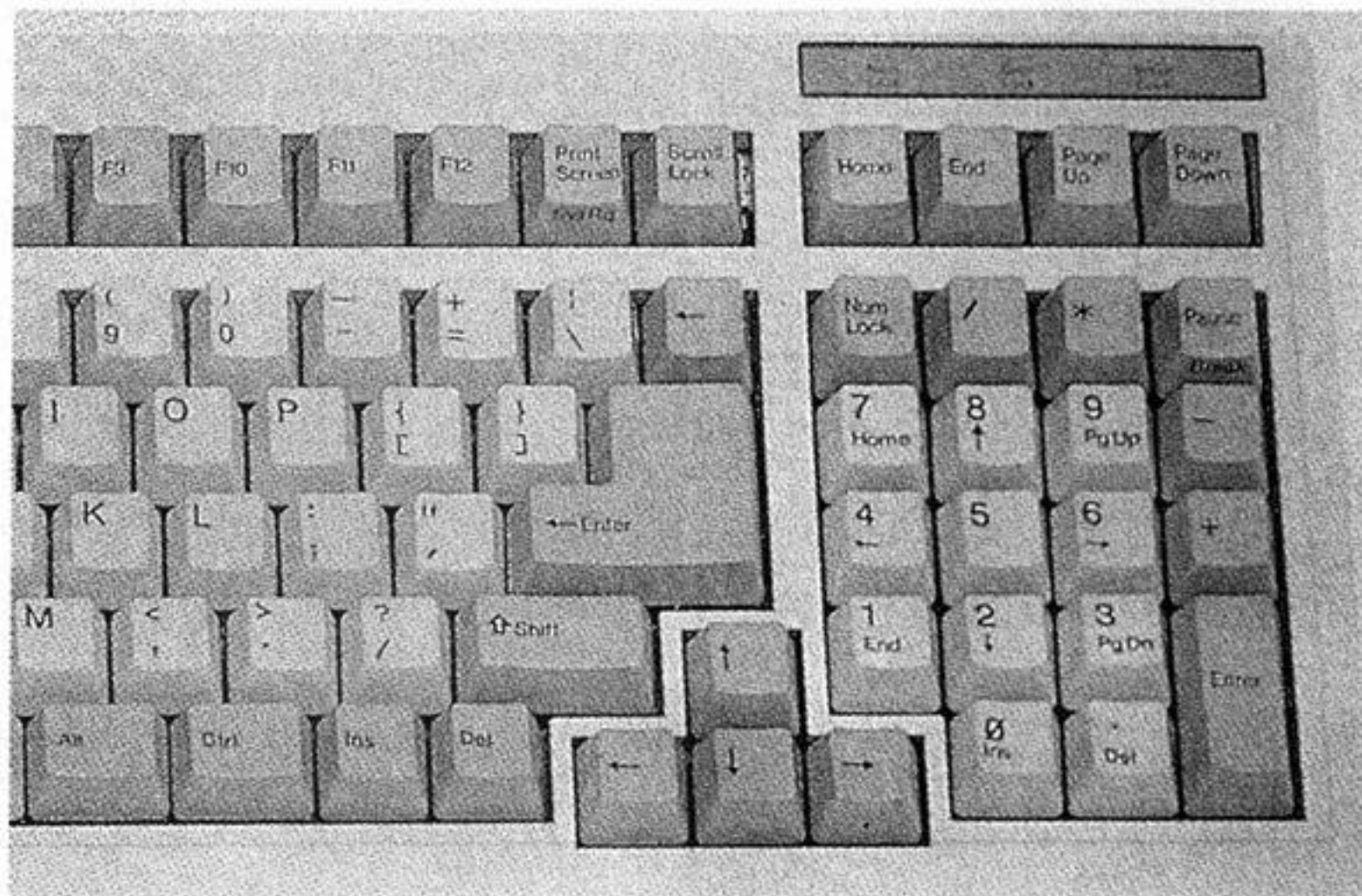
Il controllo del computer passa quindi ai due file nascosti; subito dopo, se presente, viene letto ed interpretato il file **Config.sys** (oggetto di approfondimento del presente articolo), e caricato un altro file, il famoso **Command.com**, cioè l'interprete dei comandi interni del Dos.

Quest'ultimo programma legge il file **Autoexec.bat** (se esiste), lo interpreta e lo "lancia"; altrimenti chiede la data, l'ora e visualizza la versione del Dos.

Ricapitolando quanto detto, per far funzionare un computer è necessario un supporto magnetico (sia esso floppy o hard) su cui è registrato il loader per caricare i file nascosti ed il file command.com. Il modo più semplice per creare un disco con queste caratteristiche, come è noto, è formattarlo con l'opzione **/s**.

Config.sys

Il file **Config.sys** è un file ASCII (leggi aski), contenente le informazioni necessarie per il funzionamento ottimale del computer.

Tale file viene letto **solo** in fase di bootstrap; nel caso in cui si intenda apportare modifiche, per fare in modo che il computer le riconosca saremo costretti a resettare il sistema. Il file config.sys contiene diverse informazioni, tra le più importanti: numero massimo di drive logici, quantità di memoria riservata ai buffer dei dischi, nazionalità della tastiera, numero di file apribili, device di driver ed

Escape e dintorni

Funzioni grafiche

Il carattere **X** viene considerato come un parametro. Ad esempio...

```
esc[xm
```

...oppure...

```
esc[x; ..; xm
```

...imposta gli attributi dello schermo.

Il valore di **x** ha il seguente significato:

0 tutti gli attributi vengono rimossi e il video torna alle condizioni normali
1 alta intensità
5 lampeggio
7 attiva reverse
8 spegne il display
30 primo piano nero
31 primo piano rosso
32 primo piano verde
33 primo piano giallo
34 primo piano blu, sottolineatura su video monocromatico
35 primo piano magenta
36 primo piano ciano
37 primo piano bianco
40 fondo in nero
41 fondo rosso
42 fondo verde
43 fondo giallo
44 fondo blu
45 fondo magenta
46 fondo ciano
47 fondo bianco

Per esempio, volendo il colore di primo piano **bianco** su fondo **nero** bisognerà impartire...

```
esc[37;40m
```

Per riportare le condizioni iniziali:

```
esc[0m
```

Il comando...

```
esc[=xh
```

...imposta la larghezza dello schermo, o la modalità grafica, se il valore di x varia tra 0 e 6:

0 =40 colonne per 25 linee, in bianco e nero
1 =40 colonne per 25 linee, a colori
2 =80 colonne per 25 linee, in bianco e nero
3 =80 colonne per 25 linee, a colori
4 =320 * 200, a colori
5 =320 * 200, in bianco e nero
6 =640 *200, in bianco e nero

Il comando...

```
esc[=xl
```

...ha la stessa funzione della sequenza precedente, con gli stessi parametri, eccetto per

7 giunti alla fine del rigo i caratteri successivi non sono visualizzati.

Funzioni di cancellazione

Con il comando...

```
esc[2J
```

...il video viene cancellato ed il cursore spostato il alto a sinistra.

Con...

```
esc[K
```

...viene cancellata la linea corrente a partire dalla posizione del cursore (inclusa).

Spostamento del cursore

Considerando x e y le coordinate del cursore, con...

```
esc[y;xH
```

...oppure...

```
esc[y;xf
```

...il cursore viene spostato alla posizione indicata; se i parametri non vengono specificati, il cursore viene spostato in alto a sinistra.

I comandi...

```
esc[yA
esc[yB
```

...spostano il cursore verso l'alto (verso il basso) del numero di linee specificato dal parametro; se non è specificato viene assunto il valore unitario.

I comandi...

```
esc[xC
esc[xD
```

...spostano il cursore verso destra (sinistra) del numero di colonne specificato dal parametro; se non è specificato nessun parametro viene assunto uno.

Il comando...

```
esc[6n
```

...memorizza nel buffer di tastiera le coordinate del cursore, ma genera un errore perché, per il computer, è come se le coordinate fossero digitate; non può essere quindi usato direttamente (veniva usato nei terminali per far sapere al computer principale la posizione del cursore).

Il comando...

```
esc[y;xR
```

...sposta il cursore alle coordinate indicate quando viene ricevuta la sequenza precedente, mentre con...

```
esc[s
```

...viene salvata la posizione corrente del cursore e con...

```
esc[u
```

...ripristinata la posizione del cursore.

Riassegnazione dei tasti

Un tasto viene individuato mediante un codice che lo contraddistingue dagli altri. Si tenga presente che molti tasti (tasti di funzione, tasti del cursore, etc) sono formati da una sequenza di **due** caratteri; in questo caso il primo vale sempre **0**. Negli esempi che seguono, con **t1** (eventualmente 0, come già detto) e **t2** abbiamo indicato i codici ascii del tasto da riassegnare; con **string**, la stringa da assegnare al tasto premuto, e con **asc** il codice ascii del carattere da assegnare al tasto. Con tali premesse, alcune sequenze valide sono:

```
esc[t1;t2;"string";asc;"string";ascp
esc[t1;"string";13p
```

Alcuni esempi chiariranno meglio le idee: se al tasto **F10** vogliamo far corrispondere il comando **Dir** + **Return**, la sequenza sarà...

```
esc[0;68;"dir";13p
```

Se al tasto **F9** deve corrispondere **Cls** + **Return** + **Date** + **Return**, digiteremo:

```
esc[0;67;"cls";13;"date";13p
```

...e così via.

altri file eventualmente richiesti da pacchetti di vario genere.

Device di driver

Lo scopo principale dei **device di driver** (che chiameremo semplicemente *"driver"* e significa letteralmente *dispositivi di guida*) è quello di gestire periferiche opzionali, apparecchi che, di solito, non sono così diffusi da essere forniti di serie con il computer.

I driver sono programmi molto particolari, scritti in linguaggio macchina, dotati di una struttura diversa dai normali file eseguibili; possono essere caricati solo automaticamente e **solo** in fase di bootstrap, momento in cui, come abbiamo già detto, il computer legge i vari file di configurazione, tra cui config.sys.

Tali file, quindi, non devono essere lanciati dall'utente, ma risultano sempre residenti ed attivi. I driver, di regola, sono registrati in file con estensione **.sys** e vengono caricati inserendo, nel file config.sys, righe del tipo...

```
device = nome_driver
```

A partire dalla versione 3.30 Ms - Dos sono stati aggiunti due file, **keyboard.sys** e **country.sys**, che presentano una gestione diversa, per cui non possono essere caricati come tutti gli altri driver.

Ansi.sys

L'ANSI (**A**merican **N**ational **S**tandard **I**nstitute) è un istituto americano che si occupa dei problemi relativi alla standardizzazione di prodotti e procedure, tra cui anche quelle relative all'informatica.

Prima dell'avvento dei personal computer, i grandi computer *(mainframe)* erano collegati a più terminali tramite linee; il terminale riceveva i vari messaggi, li stampava sullo schermo e inviava al computer ciò che veniva digitato su tastiera.

Tale sistema, però, si rilevò limitato perché non si potevano effettuare operazioni particolari, tra cui la cancellazione dello schermo, l'evidenziare parti di testo e così via.

Per questo motivo l'ANSI standardizzò sequenze particolari di caratteri in modo che il terminale le interpretasse non come messaggi alfanumerici da stampare, ma come **comandi** inviati al terminale stesso per effettuare le operazioni suddette.

Quando, in seguito, comparvero i primi personal computer, si pensò di utilizzarli anche come emulatori di terminali, e si provvide, pertanto, a servirsi del driver **Ansi.sys** per far in modo che il personal computer potesse gestire correttamente le sequenze Ansi già diffuse come standard.

Una sequenza Ansi è formata da **tre parti**, di cui la prima è "fissa" (formata dal carattere **Esc** e dalla **parentesi quadra** chiusa); tale coppia di caratteri, intestazione di ogni sequenza Ansi, viene chiamata **CSI** (**c**ontrol **s**equenze **i**ndicator). Segue una serie di parametri separati dal carattere di punto e virgola (;) e, per ultimo, un carattere indicante la **funzione** svolta dalla sequenza stessa.

Particolare importanza ha la lettera **p** che consente di ridefinire i tasti. Le sequenze di Ansi, poiché iniziano sempre con il carattere Esc, vengono chiamate anche sequenze di **esc**ape.

Ricordate che una sequenza di Ansi, per essere riconosciuta come tale dal computer, deve sempre essere "lanciata" direttamente dal Dos, o con programmi che usano le routine di output del Dos, senza alterare l'attributo maiuscolo/minuscolo del carattere di coda e senza spazi intermedi.

Il miglior sistema per stampare le sequenze di Ansi consiste nel registrarle su un file sequenziale per poi attivarle con il comando **type** come se, cioè, lo si volesse visualizzare su schermo.

In alternativa è possibile modificare il Prompt del Dos per ottenere effetti particolari ogni volta che il Dos stampa il Prompt.

```
10 OPEN "i", #1, "menu.arc"
20 LINE INPUT #1, S$: OPEN "o", #2, S$ + "\menu.ans"
30 OPEN "o", #3, S$ + "\menu.scr": PRINT #3, CHR$(27); "[2J"
40 PRINT #2, CHR$(27); "[0;94;"; CHR$(34); "type "; S$;
"\menu.scr"; CHR$(34); ";13p"
50 I = 1
60 IF I > 10 THEN END
70 LINE INPUT #1, A$: PRINT #3, USING "## \
\"; I; A$
80 PRINT #2, CHR$(27); "[0;"; : PRINT #2, USING "##"; I + 83;
90 LINE INPUT #1, C$: IF C$ = "*" THEN 110
100 PRINT #2, ";"; CHR$(34); C$; CHR$(34); ";13"; : GOTO 90
110 PRINT #2, "p": IF EOF(1) THEN END
120 I = I + 1: GOTO 60
```

· **Il semplice programma in Gw - Basic**

Il comando Prompt

Con **Prompt** si intende il messaggio che il computer stampa per far sapere all'operatore di essere in attesa di nuovi ordini.

Per default il Prompt è formato dalla lettera indicante il drive corrente e dal carattere di maggiore (**>**); il Dos, comunque, mette a disposizione, appunto, il comando Prompt che consente di modificare il Prompt originale del Dos fornendo altre informazioni.

Il suo utilizzo è molto semplice; basta, da Dos, impartire il comando Prompt seguito dai caratteri e **metacaratteri** desiderati (cioè caratteri dotati di significato particolare per il computer), in accordo con la tabella dei metacaratteri qui riportata, che devono essere preceduti dal carattere dollaro ($).

**$** carattere "$"
t ora corrente
d data corrente
p drive e subdirectory correnti
v numero di versione
n drive corrente
g carattere ">"
l carattere "<"
b carattere "|"
s spazio (ammesso **solo** all'inizio)
h backspace
e escape

q carattere "="

Un esempio chiarirà meglio le idee. Provate a digitare (su di un **unico rigo** e **non** su due righe, come qui di seguito riportato per esigenze di impaginazione) e premete il tasto Return...

```
Prompt $e[s$e[A$e[K$e[1;
   67H$d$e[2;67H$t$e[u$p$g
```

Esaminiamo quale sarà il Prompt del Dos dopo aver impartito il comando.

Vengono stampate delle sequenze di Ansi che svolgono le seguenti funzioni:

* memorizzano la posizione del cursore;
* il cursore viene fatto salire di un rigo;
* viene cancellato il rigo corrente;
* il cursore viene portato alla riga 1 colonna 67 e con un...
* ...metacarattere (**d**) viene stampata la data;
* il cursore viene spostato alla riga 2 colonna 67 e viene stampata l'ora;
* quindi viene ripristinata la posizione iniziale del cursore e con i metacaratteri (**pg**) vengono stampati drive e subdirectory correnti e il carattere maggiore.

Il programma dimostrativo

Il programma riportato come esempio (ANSI.BAS) consente di attivare altri programmi residenti su disco rigido mediante la semplice pressione dei tasti funzione e del tasto Shift (invece di esser costretti a digitare intere sequenze di tasti per "entrare" in subdirectory, attivare il programma che interessa, eccetera).

Il programma Basic contiene tutte le istruzioni per generare i vari file di "appoggio" di cui il computer si servirà, una volta resettato e lanciato nuovamente.

La procedura, infatti, richiede, per funzionare correttamente, che il driver **Ansi.sys** sia residente in memoria e la presenza di un file, scritto secondo uno standard definito, che tra breve descriveremo riga per riga.

Il file Ascii di cui parliamo (**menu.arc**) deve essere formato in modo tale da rispondere adeguatamente alle esigenze del programma Basic pubblicato.

Ciò significa che dovrete, a parte, scrivere un banale file Ascii prima di lanciare il programma Basic (che, lo ripetiamo, provvederà a generare i file idonei a rendere, in seguito, del tutto automatica l'intera procedura).

La prima linea del file Menu.arc, di cui stiamo parlando, deve rappresentare il nome della subdirectory (che, nel caso specifico dell'esempio che seguirà, abbiamo chiamato semplicemente **ansi**) in cui saranno scritti (automaticamente dal programma Basic) altri due file; il file Menu.arc deve quindi contenere altre righe che rappresentano (ad esempio) il nome del programma da lanciare (**WordStar 4.00**); la sequenza di tutti i comandi che normalmente verrebbero digitati se si operasse da tastiera per lanciare il programma (**cd \ws4, ws, cd** \); ed infine un asterisco (*) che ha il compito di indicare (al programma Basic) altre righe che consentiranno al computer di attivare un altro programma (nel nostro caso **QuickBasic**). E così via fino ad un massimo di **10** programmi (valore che, volendo, può essere modificato dal lettore).

Riepilogando: se sull'hard-disk sono presenti **WordStar 4.00** e **QuickBasic 4.50**, e si vogliono creare i file nella subdirectory di nome ansi, il file menu.arc sarà così composto (e digitato utilizzando un comune editor Ascii):

```
c:\ansi
WordStar 4.00
cd \ws4
ws
cd \
*
QuickBasic 4.50
cd \qb45
qb
cd \
*
```

I file creati nella subdirectory ansi avranno (in seguito) nome **Menu.scr** e **Menu.ans**. Il primo conterrà un semplice menu (magari personalizzabile dopo aver bene compreso il funzionamento della procedura); il secondo, invece, le sequenze di Ansi per le riassegnazioni dei tasti. In particolare, la prima riga fa in modo che, premendo i tasti **Ctrl + F1**, viene stampato sullo schermo il file Menu.scr. I due file, lo ripetiamo, vengono generati automaticamente dal programma Basic di queste pagine.

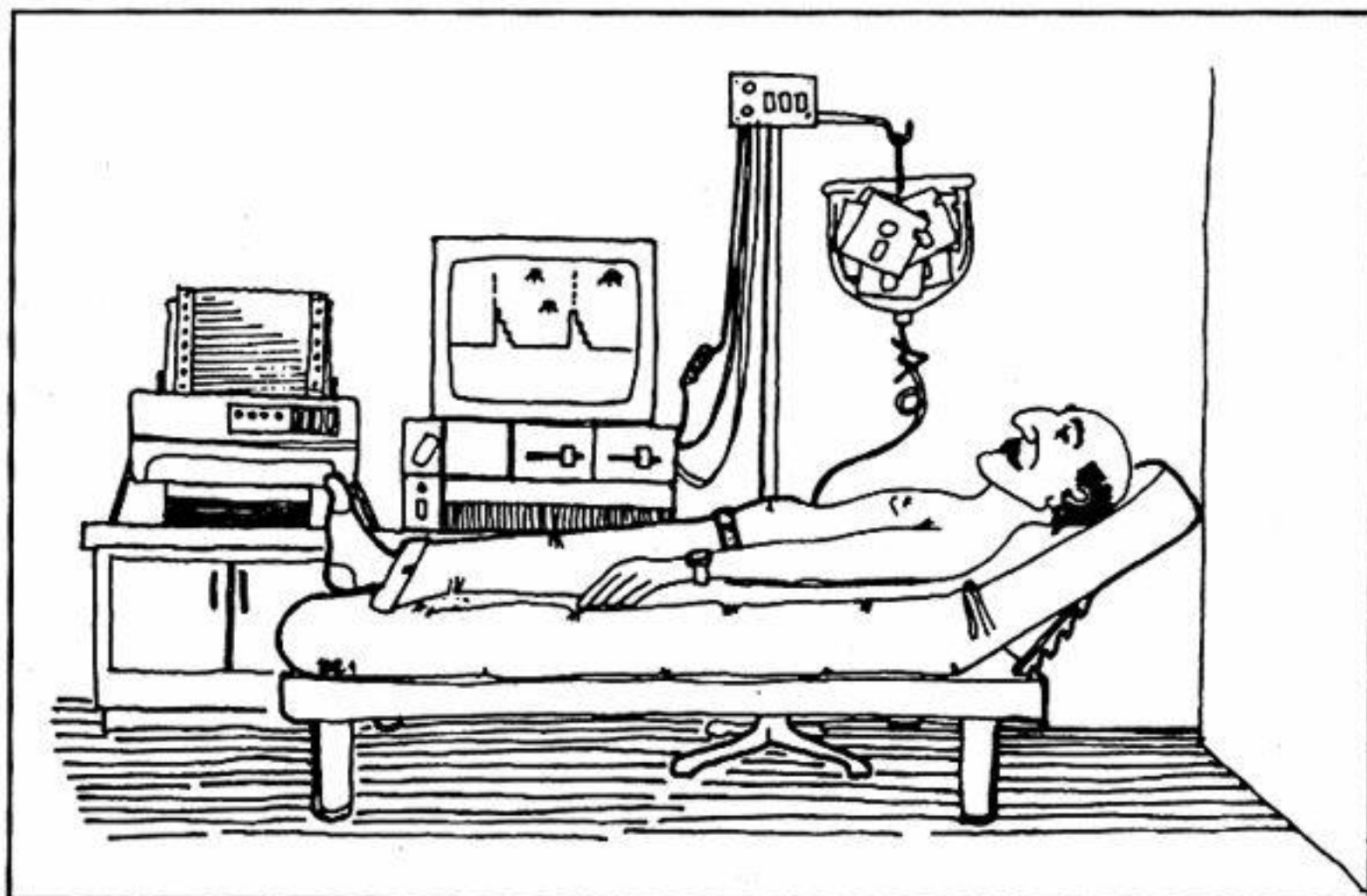
Le sequenze di ansi, a questo punto delle operazioni, non sono ancora state attivate; con il comando type, quindi, verrà visualizzato il file menu.ans. Premendo i tasti **Ctrl** + **F1** verrà infatti visualizzato il menu...

1 WordStar 4.00
2 QuickBasic 4.50

...mentre con i tasti **Shift** + **F1** verrà lanciato WordStar e con i tasti **Shift** + **F2**, invece, il QuickBasic.

Come funziona

Il programma legge il file menu.arc e, in base a questo, crea altri due file.

Per prima cosa apre in input il file **menu.arc**, individua la subdirectory in cui

trascrivere i file menu.scr e menu.ans e li crea.

Nel file **menu.scr** viene trascritta la sequenza di ansi che cancella lo schermo; mentre nel file **menu.ans** viene trascritta la sequenza di ansi che (al momento in cui verranno premuti i tasti Ctrl e F1) visualizzerà il file menu.scr mediante l'attivazione automatica del comando type.

Quindi viene posta a 1 la variabile **i**, incaricata di conteggiare i programmi inseriti nel menu. In seguito il programma controlla se la variabile **i** è maggiore di 10 e, in tal caso, il programma termina, altrimenti viene letta la riga successiva del file menu.arc (cioè il nome del programma da inserire nel menu) che viene subito trascritto nel file menu.scr.

In seguito viene riversata, nel file menu.ans, la sequenza di ansi che ridefinisce il tasto da far corrispondere al programma da lanciare; poi vengono lette altre righe dal file menu.arc (fino all'asterisco) ove sono riportati tutti i comandi necessari per lanciare il programma desiderato.

Quando viene incontrato l'asterisco, viene riportato (sempre nel file menu.ans) il carattere **p**: la sequenza di ansi, per la definizione del tasto, è finalmente completa; rimane solo da controllare se il file menu.arc è terminato; in caso contrario la variabile **i** viene incrementata e l'elaborazione ripete il ciclo finché il valore di **i** raggiunge 10.

Le variabili usate nel listato sono:

S$** (subdirectory dove creare i file menu.scr e menu.ans); **A$ (nome del programma da lanciare o messaggio da far apparire nel menu); **C$** (comandi idonei per eseguire i programmi inseriti nel menu).

Commenti al programma

10 apre in input il file menu.arc;

20 legge la subdirectory dove creare i file menu.arc e menu.ans e crea il file menu.ans;

30 crea il file menu.scr e vi trascrive la sequenza di ansi per cancellare lo schermo.

40 scrive nel file menu.ans la sequenza di ansi per stampare sul video con il comando type il file menu.scr quando verranno premuti i tasti Ctrl e F1.

50 setta a 1 il contatore per i programmi inseriti nel menu.

60 controlla se nel menu sono stati inseriti più di 10 programmi e in questo caso termina.

70 legge il nome del programma da inserire nel menu e lo scrive nel file menu.scr.

80 scrive nel file menu.ans la sequenza per la ridefinizione del tasto corrispondente al programma da lanciare.

90 legge un comando per lanciare il programma da menu, se è un asterisco va alla linea 110.

100 scrive il comando letto nella linea precedente nel file menu.ans e torna alla linea 90.

110 scrive nel file menu.ans la lettera p, chiudendo la sequenza di ansi per la definizione del tasto, quindi controlla se il file menu.arc è finito e in questo caso il programma termina.

120 incrementa il contatore dei programmi inseriti nel menu e va alla linea 60.

Consigli e avvertenze

La memoria riservata alla ridefinizione dei tasti è limitata; è meglio quindi ridurre allo stretto indispensabile i comandi necessari per lanciare un programma (cambio directory, attivazione programma, ritorno alla directory principale e così via).

Se si vuole ridefinire un tasto già definito precedentemente con una nuova sequenza di ansi, è fortemente consigliato resettare prima il computer per evitare indesiderate sovrapposizioni.

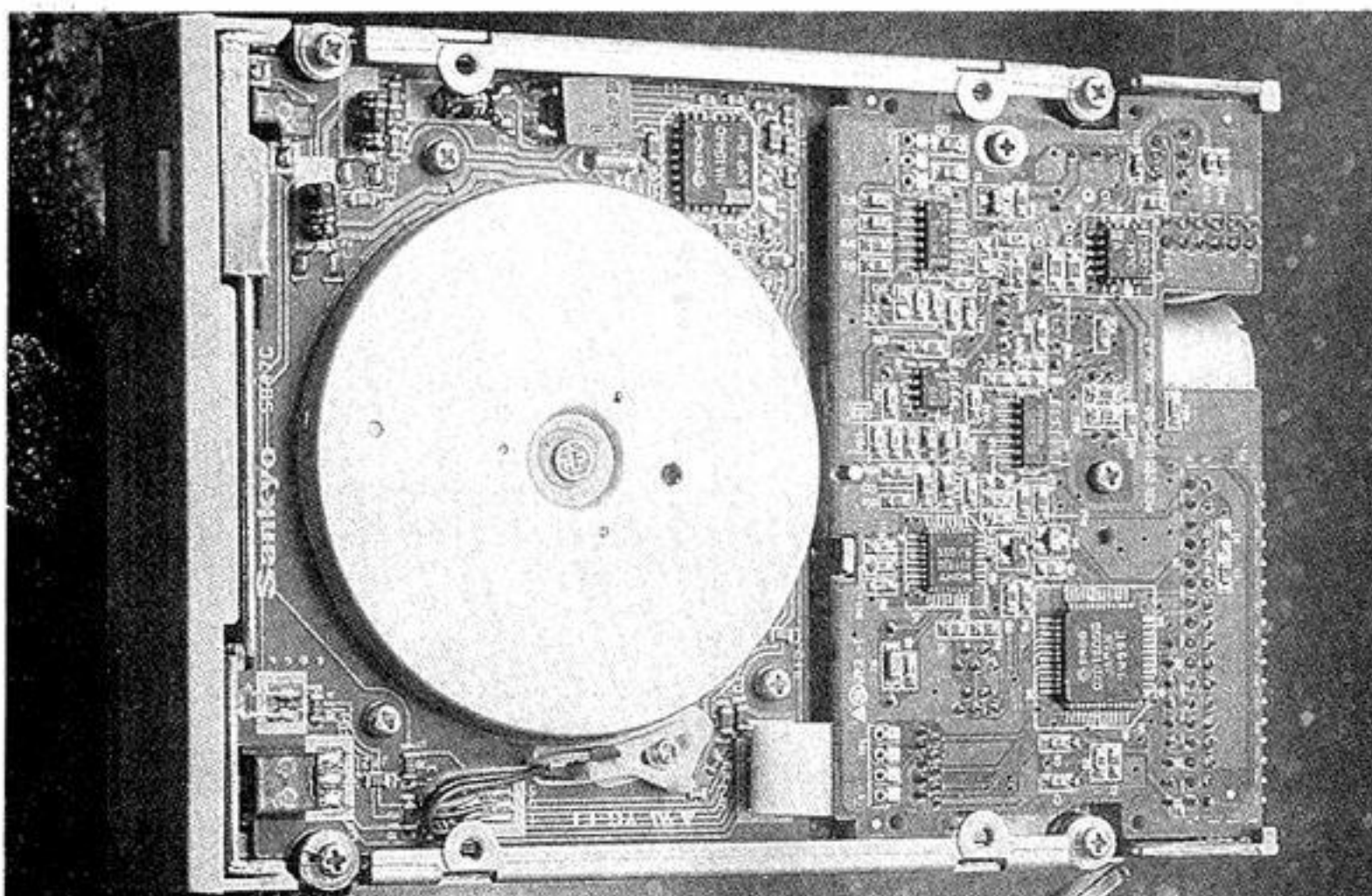
Può essere interessante esaminare, con il comando type, i file menu.ans e menu.arc **prima** di installare ansi.sys. Dato che ci siete, esaminate il contenuto di **Config.Sys**; è probabile che, in questo file, non sia presente il comando `device = ansi.sys`.

La procedura più semplice per installare in memoria il drive ansi.sys (se, appunto, non risulta installato) è la seguente: riportare (mediante una semplice operazione di copia) nella directory principale il file ansi.sys e digitare direttamente, da Dos, il comando...

```
echo device=ansi.sys > c:\config.sys
```

In questo modo la linea `device = ansi.sys` sarà inserita "in coda" al file config.sys, e il device di driver verrà caricato ogni volta che si accende il computer.

Se si vogliono creare i file menu.ans e menu.arc nella directory principale del disco rigido, la prima linea del file menu.arc dovrà essere **c:** e non **c:** come qualcuno potrebbe giustamente(?) pensare (vedere linee 20 e 30).

Non dimenticate che l'**ultima** linea del file menu.arc **deve** essere un asterisco. Il programma non crea la subdirectory indicata nella prima linea del file menu.arc, dal momento che si suppone già esistente.

di Giuseppe Maino e Vincenzo Piazza

Le divisioni; due risposte alla sfida matematica

La sfida lanciata sul n. 82 ha avuto molti consensi; due nostri lettori, partendo dalla stessa ipotesi, propongono lavori originali in C e in Basic

Due lettori, tra quelli che hanno accettato la sfida, hanno inviato un lavoro decisamente interessante sia perchè l'intera procedura di divisione può "girare" su un qualsiasi computer (**Amiga, Ms - Dos, C/64**) sia perchè la notevole brevità dei quattro listati (due in **Basic** ed altrettanti in **C**) sarà validissima per chi intende affrontare il **C** un po' per volta, aiutandosi -ed è questo il caso- della corrispondente "traduzione" nel più rassicurante interprete Basic.

Tale particolare dimostra ancora, se mai ce ne fosse stato bisogno, che i linguaggi compilatori non sono poi così "tremendi" come si immagina.

Basta solo un po' di pazienza, un po' di impegno e (scusate l'immodestia) l'indispensabile aiuto dei lettori di Commodore Computer Club!

Giuseppe Miano (Catona - RC)

Se in una divisione le trasformazioni del dividendo si ripetono periodicamente (vedi esempio a parte), si è in presenza, appunto, di un periodo.

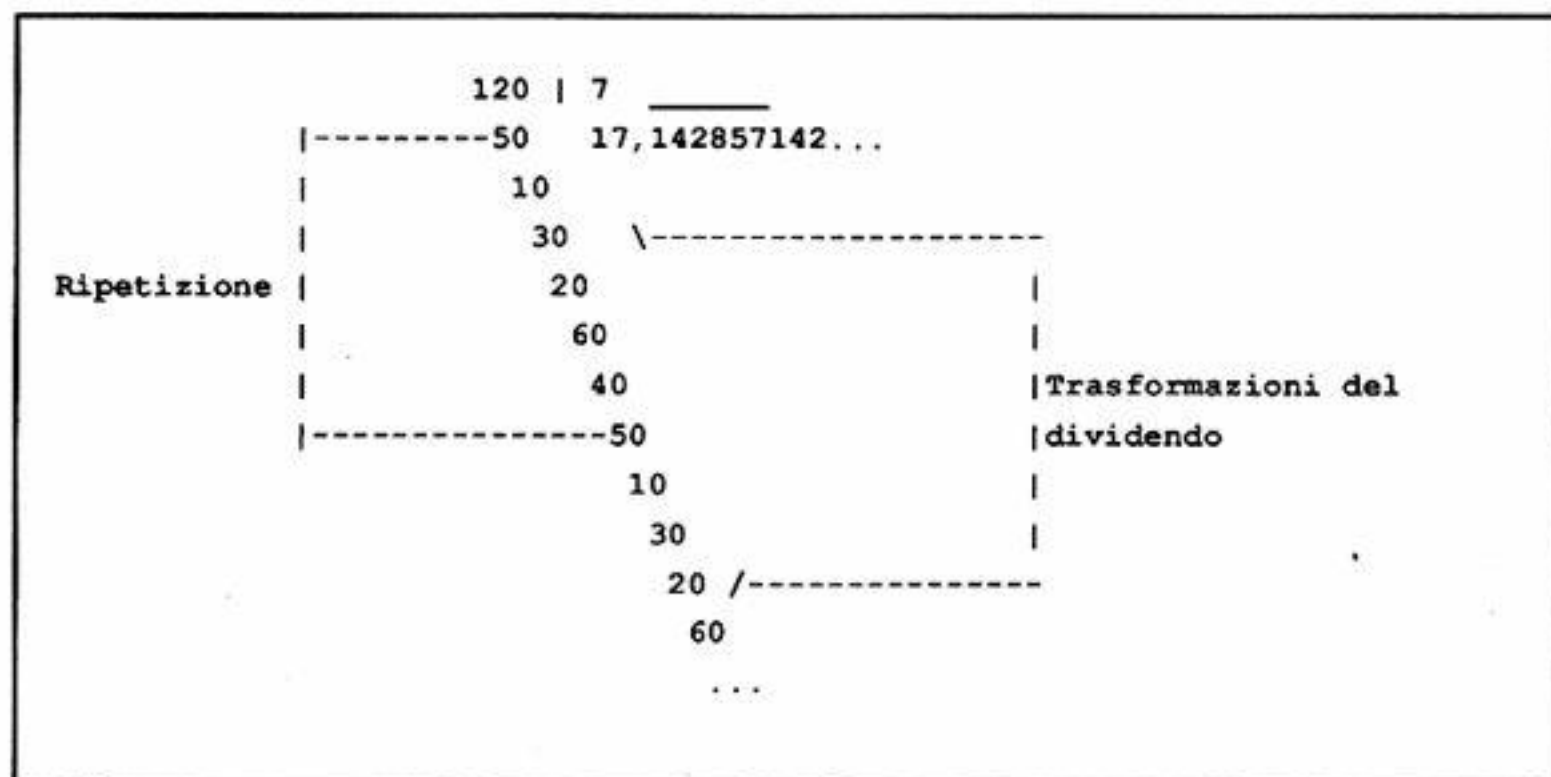
Per verificare, in un quoziente, la presenza di un **periodo**, è sufficiente eseguire la divisione memorizzando, in un'apposita matrice, le trasformazioni del dividendo ottenute, e controllare ogni volta che quella particolare trasformazione non sia già contenuta fra quelle precedenti.

L'individuazione di due trasformazioni uguali indica la presenza di un periodo che può essere visualizzato, mediante semplici operazioni, così come la sua **lunghezza**, la sua **posizione** all'interno del quoziente, l'**antiperiodo** e le cifre intere.

Il programma

Il programma richiede in sequenza il numeratore, il denominatore ed un valore chiamato **Fattore V**, valore contenuto nella variabile **S**.

Il Fattore V, che deve valere almeno **2**, sta a significare quante cifre della matrice il computer deve verificare. O meglio: dal momento che, spesso, il periodo inizia non molto dopo la prima cifra decimale, il programma fa in modo che (dopo aver individuato l'Nesima trasformazione) non la confronti con le precedenti **n-1**, ma soltanto con le prime **s**.

In tal modo si risparmia moltissimo tempo, specialmente quando siamo in presenza di un periodo lungo. E' affidato alla discrezione dell'utente stabilire il numero s che ritiene più opportuno.

Schema tradizionale di una divisione

Un "periodo" viene individuato dalla eventuale ripetizione ciclica dei resti che si ottengono eseguendo una qualsiasi divisione

Se, poi, non si ha proprio idea quale sia la prima cifra decimale da cui inizia il periodo, basta inserire un valore molto alto (**10000 / 15000**, comunque superiore alle dimensioni della matrice **j()**). Di solito basta assegnare **s = 10** per ottenere una discreta velocità di elaborazione.

Dimensionando, ad esempio, la variabile j() con il valore di 10000 (compatibilmente, comunque, con la memoria del computer) è possibile stabilire la massima lunghezza del periodo. Nel programma pubblicato il valore viene fissato, appunto, a 10000 cifre.

In seguito il programma esegue una banale routine per individuare le varie cifre decimali, mentre allo stesso tempo controlla che vi sia un periodo. Quando (e se) lo trova, salta alla linea 210, in cui vi sono i vari comandi di stampa. Vengono, a questo punto, riportati la lunghezza del periodo e le cifre da cui inizia; viene quindi attivata la stampa delle cifre intere, l'antiperiodo e, dopo la pressione di un tasto, l'intero periodo con un ritorno del carrello ogni 70 cifre.

Se, ad esempio, eseguiamo **120** diviso **7** (con Fattore V = **10**) sullo schermo dovrebbe comparire...

Numeratore **120**
Denominatore **7**
Fattore V **10**
Lunghezza periodo : **6**
Cifra dec. d'inizio : **1**
Cifre intere: **17**,
Antiperiodo :
Periodo: (Premi un tasto)
142857 (Premi un tasto)

Vincenzo Piazza (V. S. Giovanni)

La sfida consisteva nello scrivere un programma che eseguisse la divisione tra due numeri dati e che, nel contempo, individuasse l'eventuale periodo.

Il programma apparso sul n. 82 presentava, però, un piccolo errore che si ripercoteva (soprattutto nel caso di quozienti con periodo molto lungo) sulla ricerca della sequenza periodica.

Il programma "Divisioni" (sia nella versione in **AmigaBASIC** che in linguaggio **C**), che è stato scritto su un Amiga 1000, elimina l'errore ricercando il periodo in modo diverso.

Quando si esegue un rapporto, se i resti ottenuti durante la divisione si ripetono, si è sicuramente in presenza di un periodo; grazie a questo particolare, il programma presenta una routine che provvede a memorizzare in un array (in una stringa la versione in AmigaBASIC) tutti i resti che, a mano a mano, ottiene e li confronta l'uno con l'altro.

Questo metodo, oltre ad essere più sicuro, consente di individuare il periodo dopo aver calcolato un numero minore di cifre rispetto al programma apparso sulla rivista; è infatti sufficiente un numero di cifre pari alla lunghezza del periodo stesso più una.

I due programmi funzionano, quindi, in modo diverso dal programma pubblicato in precedenza. Essi, infatti, verificano la presenza di un periodo mentre eseguono la divisione e si fermano automaticamente una volta trovato il periodo stesso o raggiunto il numero di cifre decimali richiesto dall'utente.

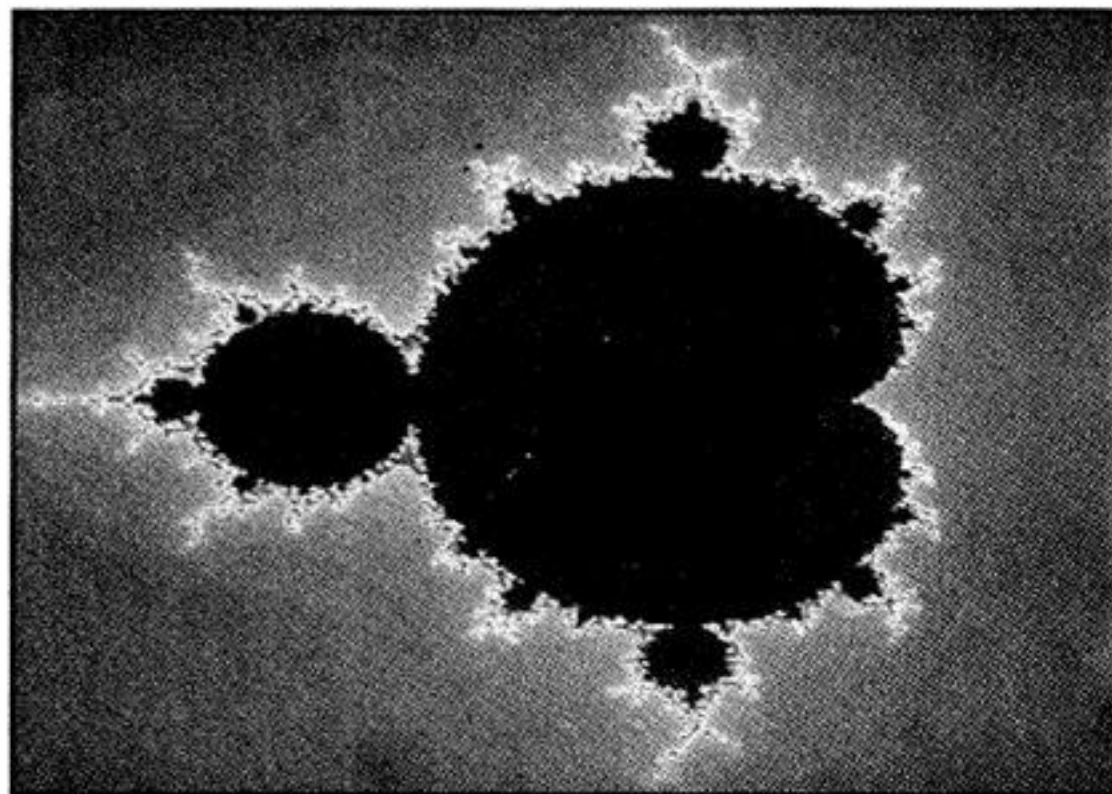
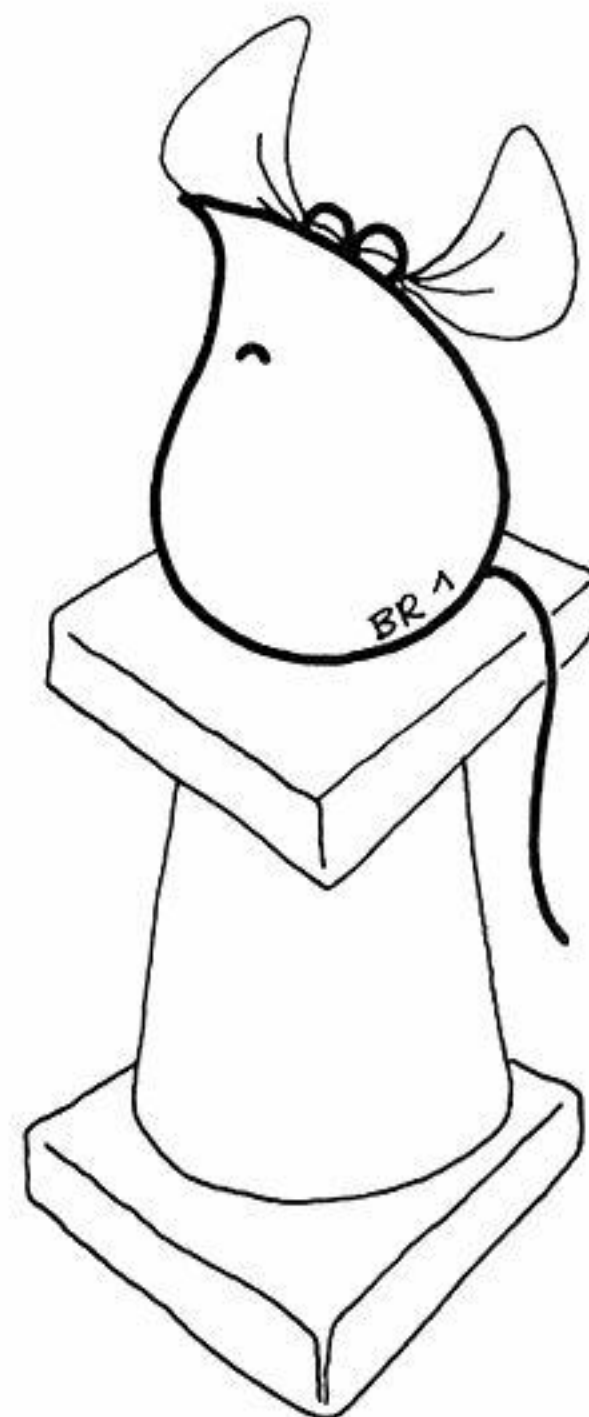
Una curiosità: **il tempo impiegato** dal programma scritto in **C**, per calcolare un certo numero di cifre, aumenta con il quadrato del numero di cifre; più precisamente la relazione che lega tempo (**t**) e cifre (**c**) è:

t = (c^2) / 83333.

A proposito di "esclusi"

La sfida lanciata nel "lontano" n. 82 ha suscitato un certo interesse; i due lavori che qui pubblichiamo, quindi, rappresentano solo una parte delle fatiche dei numerosi lettori che si sono cimentati.

La selezione, infatti, ha tenuto conto della doppia versione inviata (**Basic** e **C**) e, soprattutto, della notevole **brevità** dei listati stessi; peccato che il Pascal non abbia molti sostenitori...

Ringraziamo, comunque, tutti i lettori che, prima di inviare proprie proposte, hanno telefonato per chiedere conferma; a proposito: alcune selezioni sono dovute (quasi) esclusivamente alla data tardiva delle telefonate in oggetto; all'inizio di Maggio continuavano ad arrivare proposte in tal senso! Un consiglio è d'obbligo: datevi da fare sùbito, **appena la rivista esce in edicola**; chi ha tempo non aspetti tempo; chi inizia è a metà dell'opra; il buongiorno si vede dal mattino (ma questo che c'entra?...)

```
90 REM Programma di Giuseppe Miano
100 DEFINT k, s: CLEAR, 50000&
110 INPUT "  Numeratore   ";  a
120 INPUT "  Denominatore ";  b
130 INPUT "  Fattore V    ";  s
140 DIM j (10000): j (0)  = a\b
150 k = k + 1: a = 10* (a-b* (a\b) ): j (k)  = a
160 IF k <s THEN l = k-1 ELSE l = s-1
170 FOR t = 1 TO l
180   IF k> 1 AND j (t)  = a THEN 210
190 NEXT t
200 GOTO 150
210 PRINT: PRINT "Lunghezza periodo:  ";  k-t;
220 PRINT "       Cifra dec. d'inizio n.";  t
230 PRINT: PRINT "Cifre intere:": PRINT  j (0);  ","
240 PRINT: PRINT "Antiperiodo:"
250 FOR n = 1 TO t-1: PRINT USING "#";  j (n) \b;: NEXT n
260 PRINT: PRINT: PRINT "Periodo:";
270 PRINT TAB  (30)  "Premi un tasto"
280 WHILE INKEY$ = "": WEND:
290 FOR n = t TO k-1
300  PRINT USING "#";  j (n) \b;
310  IF  (n-t + 1)  MOD 70 = 0 THEN PRINT
320 NEXT n
330 PRINT: PRINT  TAB  (30)  "Premi un tasto"
340 WHILE INKEY$ = "": WEND: CLS: GOTO 100
```

```
10 REM DIVISIONE INFINITA - Programma di VINCENZO PIAZZA
20 WIDTH 77
30 INPUT "Dividendo ", d0
40 INPUT "Divisore  ", d1
50 INPUT "Numero di cifre decimali volute  (0 per infinite)  ", max
60 d = INT (d0/d1): PRINT: PRINT STR$ (d); ",";
70 r =  (d0-d*d1) *10
80 IF r = 0 THEN END
90 IF INKEY$ <> "" THEN PRINT "...": END
100 r1$ = STR$ (r/10)
110 i = INSTR (r$, r1$ + " ")
120 IF i <> 0 THEN r$ = r$ + r1$: PRINT: PRINT: GOTO 190
130 r$ = r$ + r1$: cifre = cifre + 1: IF cifre = max + 1 AND max> 0 THEN END
140 IF r <d1 THEN n$ = n$ + "0": PRINT "0";: r = r*10: GOTO 80
150 q = INT (r/d1): c$ = RIGHT$ (STR$ (q), 1)
160 n$ = n$ + c$: PRINT c$;
170 r =  (r-q*d1) *10
180 GOTO 80
190 nr = 0: FOR n = 1 TO i: IF MID$ (r$, n, 1)  = " " THEN nr = nr + 1
200 NEXT n
210 FOR n = nr TO cifre-1
220 IF MID$ (n$, n, 1)  <> MID$ (n$, n + 1, 1)  THEN GOTO 250
230 NEXT n
240 LOCATE 5, 2 + nr + LEN (STR$ (d) ): PRINT SPACE$ (cifre-nr): cifre = nr
250 PRINT "Il periodo inizia dalla cifra decimale no. "; nr
260 PRINT "finisce alla cifra                       no. "; cifre
270 nc = cifre-nr + 1
280 PRINT "ed e' composto quindi da "; nc; " cifre:  "; MID$ (n$, nr, nc)
```

```
/*  Programma di Giuseppe Miano */
Main ()
{
       int k, s, b, a, h, d, j[20000], l, t, n;
       k = 0;
       printf  ("  Nominatore     ");
       scanf  ("%d", &a);
       printf  ("  Denominatore  ");
       scanf  ("%d", &b);
       printf  ("  Fattore V     ");
       scanf  ("%d", &s);
       j[0] = a/b;
       cc: ;
       k +  +;
       d = a/b;
       a = 10* (a-b*d);
       j[k] = a;
       if  (k <s)  l = k-1;
       else l = s-1;
       for  (t = 0; t < = l; t +  + )  {
              if  (k> l && j[t] =  = a)  goto dd;
       goto cc;
       dd: ;

      printf  ("\n");
       printf  ("Lunghezza periodo:  %d ", k-t);
       printf  ("Cifra dec. d'inizio:  %d ", t);
       printf  ("\n Cifre intere: \n");
       printf  ("%d, \n", j[0]);
       printf  ("\n Antiperiodo: \n");
       for  (n = 1; n < = t-1; n +  + )  {
              h = j[n]/b;
              printf  ("%d", h); }
       printf  ("\n\n Periodo:tasto e [Return]");
       scanf  ("%d", s);
       printf  ("\n");
       for  (n = t; n < = k-1; n +  + )  {
              h = j[n]/b;
              printf  ("%d", h);
              }
       printf  ("\n\n\n");
}
```

```
/* DIVISIONE INFINITA - Programma di Vincenzo Piazza */
#include  <stdio.h>
int d, r, r1, n, cifre = 0, nr, d0, d1, resto[5000], l = 0, l1 = 0, q, c[5000], max;
main ()   {
   printf ("Cifre decimali  (0  =  Periodo)  ");    scanf ("%d", &max);
   printf ("Divisore                        ");    scanf ("%d", &d0);
   printf ("Dividendo                       ");    scanf ("%d", &d1);

   d = d0/d1;    printf ("\n\n%d, ", d);
   r =  (d0-d*d1) *10;
rip:;    if (!r)    ex ();
   r1 = r/10;
   for (n = 0; n < = l; n++ )
      if (resto[n] == r1)
         {   resto[l + 1] = r1;    goto fine;    }
   resto[l + 1] = r1;        l++;
   if (l == 5000)    ex ();
   if (cifre == max && max)    ex ();
   cifre++;
   if (r <d1)  { l1++;  c[l1] = 0;  printf ("0");  r = r*10;  goto rip;  }
   q = r/d1;       printf ("%d", q);
   l1++;       c[l1] = q;
   if (l1 == 5000)    ex ();
   r =  (r-q*d1) *10;
   goto rip;
fine:;    nr = n;
   printf ("\n\nIl periodo e' formato da %d cifre\n", cifre-nr + 1);
   printf ("Inizia alla cifra decimale no. %d\n", nr);
   printf ("e finisce alla cifra        no. %d\n\n", cifre);
   printf ("Cifre decimali che precedono il periodo:  ");
   for (n = 1; n < = nr-1; n++ )    printf ("%d", c[n]);
   printf ("\nPeriodo:  ");
   for (n = nr; n < = cifre; n++ )    printf ("%d", c[n]);    ex ();
}
ex ()    {   printf ("\n\n");    exit (1);    }
```

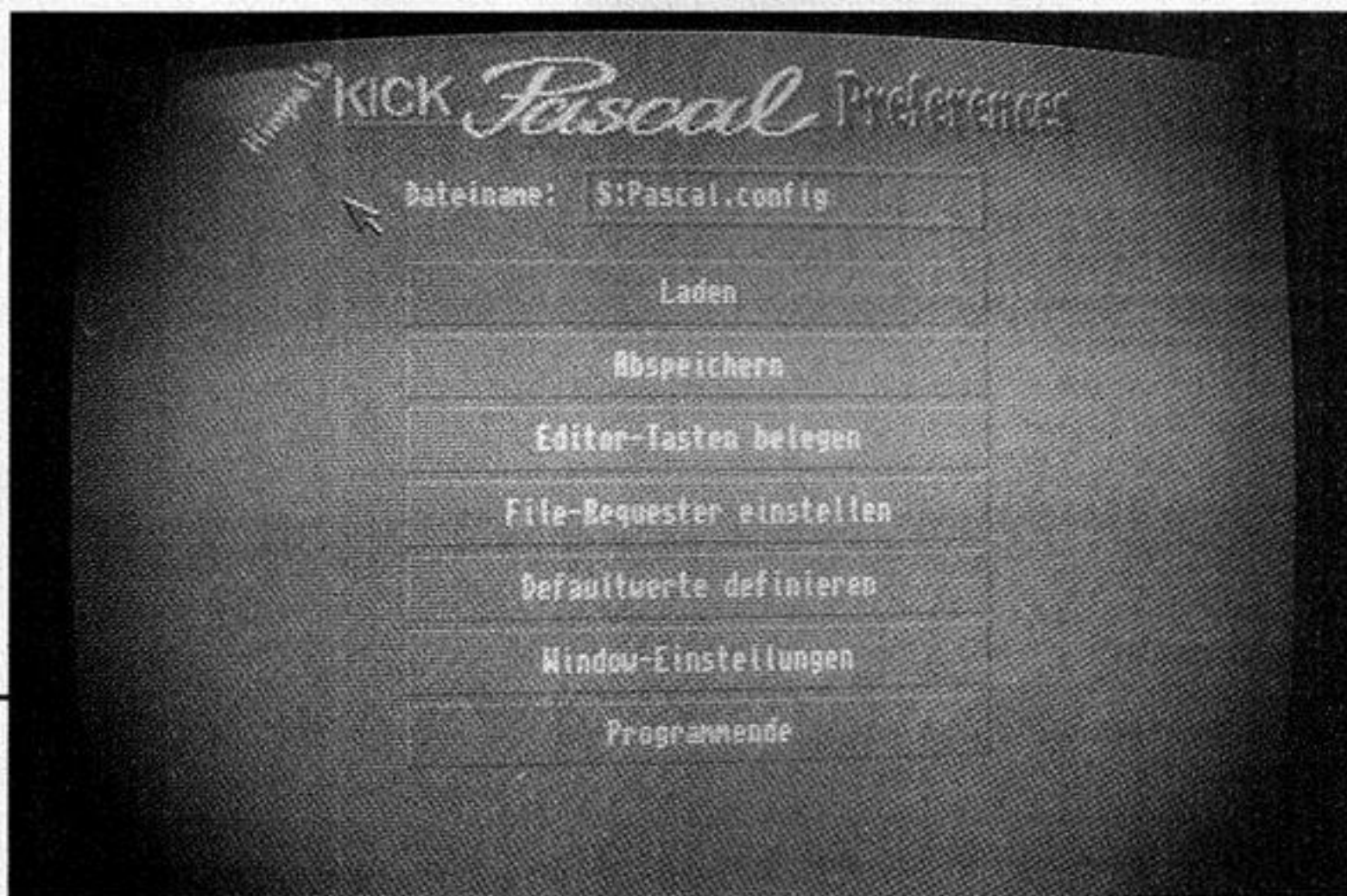
di Filippo Bosi

Amiga; finalmente un Pascal su misura

Kick Pascal è un compilatore talmente affidabile e veloce che sembra di operare su un computer Ms - Dos

Sul versante Amiga il Pascal è sempre stato trascurato a favore del "C", linguaggio più adatto ad un sistema multitasking quale AmigaDos.

Molti dei linguaggi compilatori Pascal, pur presenti sul mercato Amiga, sembravano dare ragione a questa tesi: producevano infatti file eseguibili lunghissimi e lentissimi; costringevano il programmatore a salti mortali per realizzare quello che in C era possibile con poche linee di codice e, soprattutto, un limitato tempo di compilazione.

Fino ad ora potevamo soltanto sognare un compilatore dalle prestazioni del celeberrimo **Turbo Pascal Borland**, se non fosse per il **Kick Pascal V2.0** della **HimpelSoft.** Addio lunghe attese per la compilazione, specie se **non** si possiede un Hard Disk; il linguaggio di cui ci occupiamo compila **in memoria**, proprio come il Turbo Pascal per Ms - Dos, e nulla ha da invidiare quanto ad integrazione di Editor, Compilatore e Linker.

Punti negativi

- Lentezza file Requester
- Configurazione in tedesco
- Codice di startup un po' troppo pesante

Punti positivi

- Ottima velocità di compilazione, anche su floppy-disk.
- Funziona ottimamente anche su macchine con un solo disk-drive.
- Utilizzo delle Unit tipo Turbo Pascal.
- Possibilità di adattare in breve tempo codice Turbo Pascal v5.0.
- Possibilità di adattare in breve tempo codice C.
- Ottimo supporto delle librerie di sistema (per ora v1.3).
- Compatibile con AMIGADOS/KickStart v2.0.
- Ottimi programmi dimostrativi.
- Generazione di buon codice eseguibile.
- Ottimo ambiente integrato (mai visto niente di simile su Amiga)
- Efficiente utilizzo della ram disk durante la compilazione di sorgenti.
- Adattamento del programma tramite numerose opzioni di configurazione.

Kick Pascal in sintesi

Dischetti

Tutti i file necessari per utilizzare il linguaggio sono contenuti in soli **due** dischi.

Nel **primo** trovano posto l'editor/compilatore (corredato di tutti i file *Include* necessari per la creazione di un eseguibile) mentre nel **secondo** sono presenti

i sorgenti di esempio, divisi in due directory:

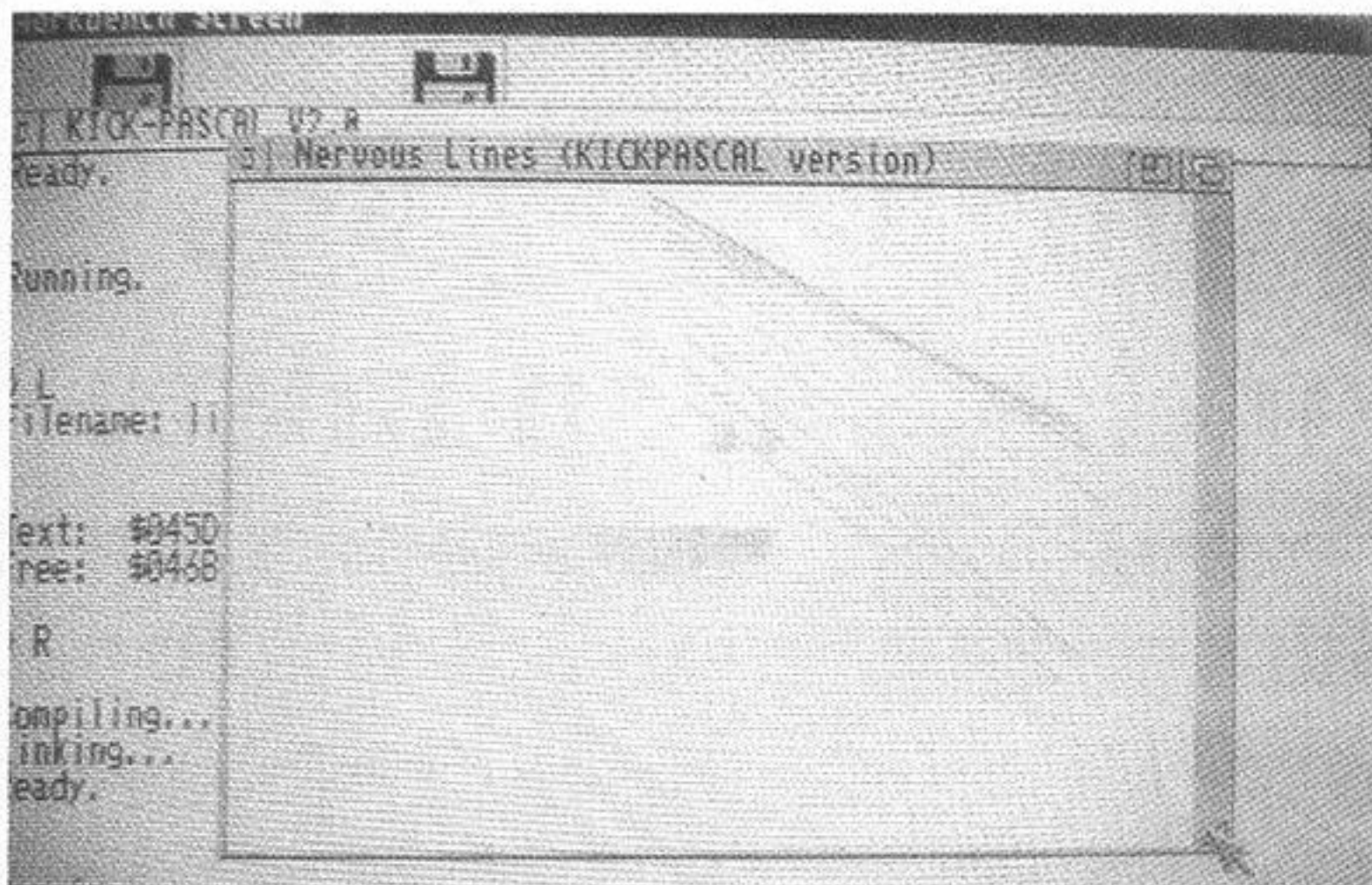
Sysprog - esempi di programmazione "a basso livello", che utilizzano funzioni specifiche delle librerie di sistema Amiga.

Demo - esempi di programmazione generici (grafica, suono, matematica, sintesi vocale, eccetera...).

Il programma può quindi essere comodamente utilizzato anche da chi possiede un Amiga con **un solo drive**, senza esser costretti a cambiare dischetto in continuazione (a meno che non si abbia bisogno di utilizzare i file sorgenti di esempio) e, come se non bastasse, i tempi di compilazione sono fantastici anche per chi non possiede un Hard Disk.

Le librerie di sistema

Uno dei punti di forza del Kick Pascal, oltre a quello di disporre di un ottimo ambiente integrato di programmazione, è la possibilità di interfacciarsi in due modi (avete letto bene...) con le librerie di Amiga: alla classica maniera del C, tramite i file **Include**, oppure utilizzando le **Unit**, files precompilati contenenti gli header di sistema (del tutto simili alle Unit note in ambiente Turbo Pascal MS - Dos) velocizzando così di molto la compilazione (vedi tabella).

Molto comodo è, inoltre, il supporto del file requester standard della **Arp Library**, accessibile molto facilmente tramite la **Unit Window** (programma Arpfrq.p). Semplice, vero? E, soprattutto, molto efficiente, date le contenute dimensioni degli eseguibili prodotti, che non necessitano di librerie run-time esterne particolari.

Se non siete pratici di Turbo Pascal, molto probabilmente non conoscete le Unit; in questo caso può essere utile esaminare con attenzione il programma **Qcdemo.p** di esempio (che utilizza, appunto, la Unit **Qcdisegna.p**) riportato in queste stesse pagine.

L'ambiente integrato

Appena lanciato, il compilatore mostra una schermata di presentazione e la richiesta dello spazio di lavoro (WorkSpace), misurato in KB, da riservare all'editor.

Qui c'è da muovere uno dei (pochi) appunti negativi al Kick Pascal: lo spazio viene sì riservato, ma in **memoria chip**(!), allocazione, questa, alquanto assurda, se si pensa che un sorgente non ha alcun bisogno di essere elaborato dai chip custom dell'Amiga.

Speriamo che a questa stranezza venga posto rimedio in una futura release del programma; per ora limitiamoci a non abbondare nel riservare memoria all'editor, se non in caso di effettiva necessità (programmi veramente complessi).

Subito dopo aver risposto al quesito, ci ritroviamo nell'editor, molto simile a quello del Turbo Pascal, anzi, talmente simile che i comandi di cancellazione, spostamento e copia di blocchi sono gli stessi.

I menu

Premendo il tasto destro del mouse si hanno a disposizione 5 menu, che qui di seguito commentiamo.

Nome Programma	Dimens. sorgente	Tempo compil. (sec.) Floppy	Hard	Dimensione file oggetto (in bytes)	
DISEGNA.P (unit)	3353 B./140 Linee	30	10	9356	(prima compilazione)
		10	8		(seconda compilazione)
		5	3	9884	(unit invece di include)
LINES.P	3538 B./180 Linee	5	5	10188	
SHOWIFF.P	20140 B./688 Linee	13	11	16534	

Tabella dei test di compilazione effettuati
Notare le differenze tra i tempi di un hard disk e quelle di un floppy

Project
E' il menu standard di tutti (o quasi) i programmi di Amiga. Da questo ambiente si caricano, salvano ed eventualmente stampano i file sorgenti contenuti nella memoria dell'editor. Notiamo la presenza delle voci **Object** ed **Exe-File**, per mezzo delle quali possiamo salvare come file oggetto (`nomefile.o`) o come eseguibile un sorgente compilato in memoria. Il requester utilizzato nell'ambiente integrato risulta un po' lento nella lettura dei file. Non si riesce a capire perché non venga utilizzato l'ottimo requester della Arp Library, dal momento che è supportato dal compilatore in modo egregio mediante una apposita Unit.

Edit
Contiene varie opzioni (tra cui **Go To**, **Find** e le altre classiche) per eseguire modifiche sul testo del sorgente.

Execute
E' il menu per mezzo del quale diamo inizio alla compilazione del sorgente presente in memoria. Sempre da qui poi possiamo farlo eseguire.

Options
Si possono scegliere sia le varie opzioni di compilazione (Test dello **Stack**, Test di **Overflow**, ecc.), che la configurazione dell'editor. Possiamo persino scegliere il linguaggio dei testi del menu: purtroppo la scelta è limitata al (solito) Inglese e al Tedesco(!).

Info
Informazioni sul programma e sui suoi autori.

Premendo il tasto **Esc**, passiamo dall'editor ad una specie di **Shell**, che ricorda molto l'ambiente del Turbo Pascal V3.x. Appare, infatti, il simbolo "**>**" in fondo alla videata, seguito dal cursore. Se a questo punto premiamo il tasto Return, appariranno, nella parte alta della pagina, alcune righe di **help** sui comandi disponibili, peraltro molto esplicative. Da qui si possono fare nè più nè meno che le cose possibili da menu, con un solo comando aggiuntivo: premendo la barra spaziatrice si ottiene una mappa abbastanza dettagliata della suddivisione in **Hunks** del codice eseguibile creato da KP e l'occupazione di memoria del file sorgente contenuto nell'editor. In questo ambiente si lavora unicamente da tastiera, rinunciando al mouse, ai menu ed ai requester per le operazioni sui files.

Configurazione

Nello stesso disco in cui è presente il compilatore, è presente il file eseguibile **Pascalprefs**, che altro non è che il programma che permette di configurare Kick Pascal al meglio, secondo le nostre esigenze.

Le opzioni presenti sono veramente moltissime, ma si verifica un solo (piccolo) problema: i messaggi che appaiono sono in **tedesco**, lingua non certo conosciuta da moltissimi utenti di computer, perlomeno non come l'inglese, che volenti o nolenti siamo stati costretti ad imparare, grazie alla quasi totale provenienza di software da paesi anglofoni.

Nella prima schermata viene richiesto in quale file salvare i dati della configurazione (per default si assume `S:Pascal.config`). Appena risposto a questa domanda, appare un menu con, nell'ordine, le seguenti opzioni:

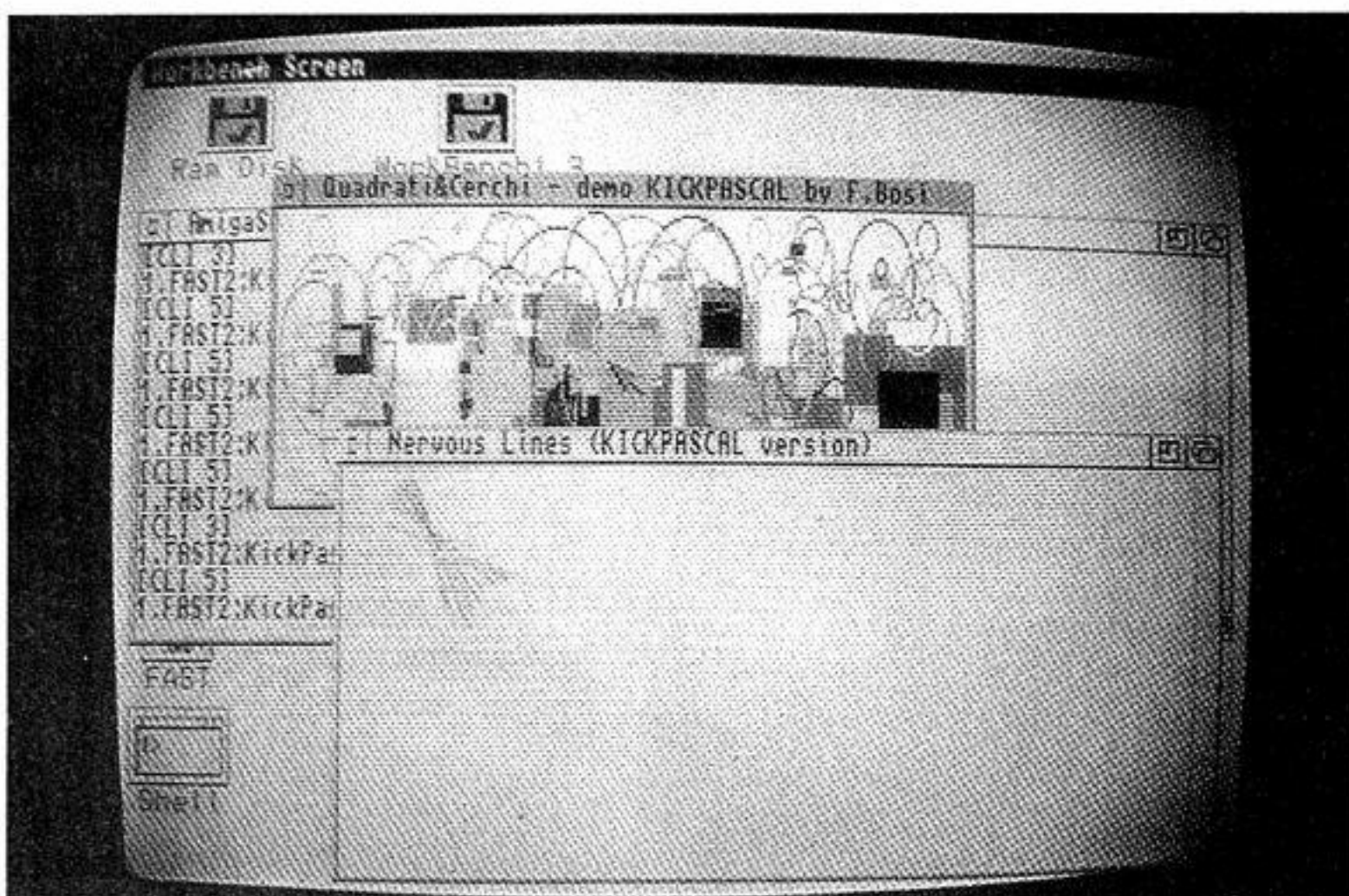
Laden = carica la configurazione dal file scelto precedentemente;
Abspeichern = memorizza la configurazione scelta;
Editor-tasten Belegen = scelta dei comandi dell'editor;
File-requester Einstellen = configurazione del file requester;
Defaultwerte Definieren = configurazione di alcuni parametri di default per il compilatore (workspace, ottimizzazioni del codice, etc.);
Window-einstellungen = configurazione dell'editor. E' possibile scegliere lo schermo di workbench per la visualizzazione dell'editor oppure uno schermo "custom"; si può anche specificare il font utilizzato, le dimensioni della finestra e perfino il suo nome;
Programmende = uscita dal programma di configurazione (attenzione a non uscire prima di aver salvato la configurazione scelta...).

Una volta scelta l'opzione desiderata, appare una schermata con le domande del caso. Sarebbe troppo lungo dare tutte le spiegazioni (o, meglio, traduzioni) delle schermate che appaiono in seguito.

Tenete presente, ad ogni modo, che torneremo prossimamente sul significato delle domande effettuate, a mano a mano che svilupperemo, su queste pagine, argomenti legati al Kick Pascal.

Il linguaggio

Molto simile al Turbo Pascal V5.0, Kick Pascal è un Pascal con molte estensioni rispetto allo standard, che per questo si sposa ottimamente con Amiga.

Risulta di facile apprendimento per chi conosce già un linguaggio come AmigaBasic e desidera sfruttare a fondo le capacità della macchina senza dovere fare salti mortali (leggi chiamate di libreria in

Basic) oppure imparare il C (croce e delizia dei programmatori).

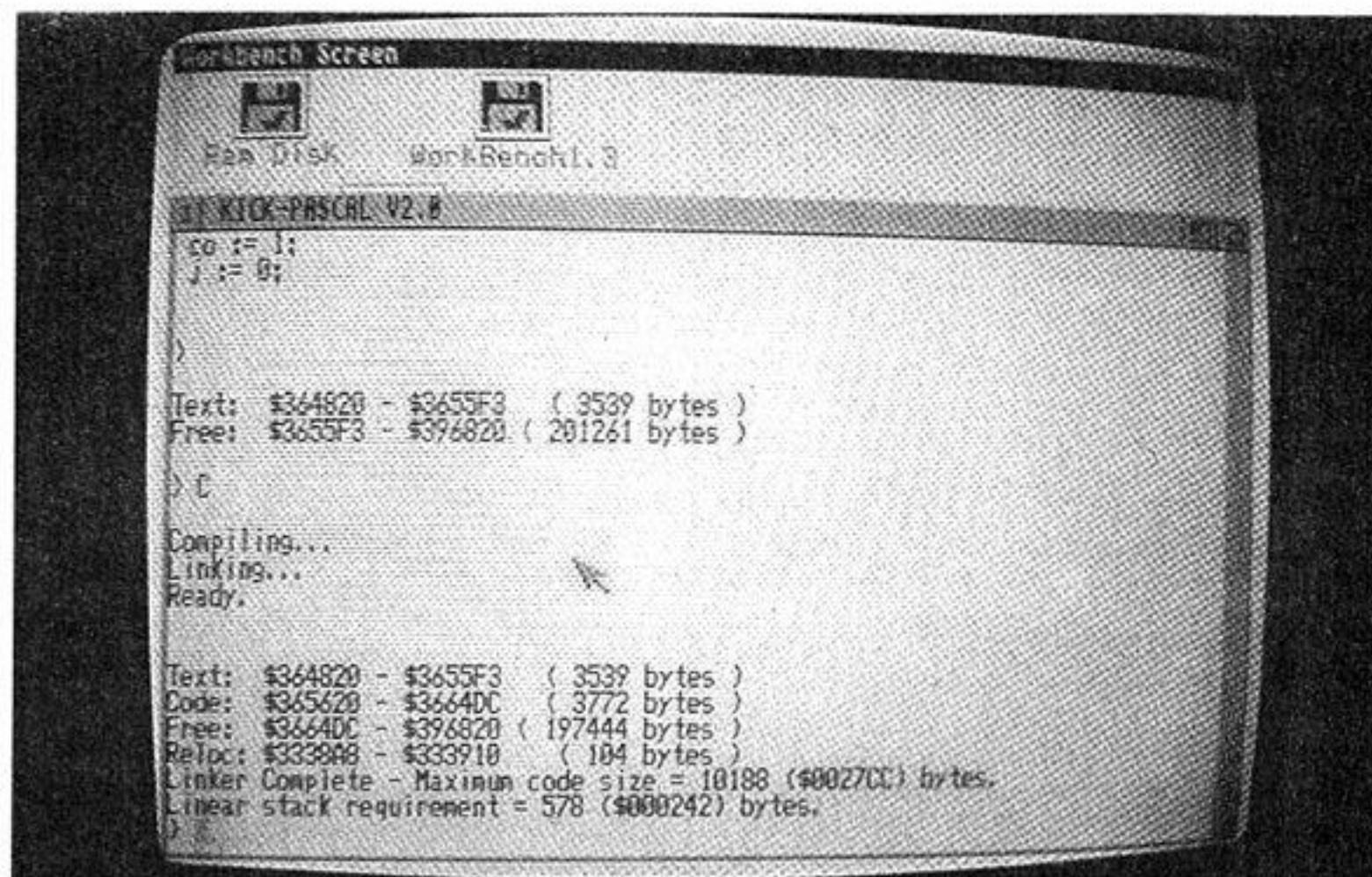
KP è ottimo per chi, provenendo da Ms - Dos, conosce il Pascal (Borland o MicroSoft). Purtroppo non utilizza, per l'esecuzione di istruzioni grafiche, i comandi **BGI** (**B**orland **G**raphics **I**nterface), ma non dovrebbe essere essere una grossa difficoltà costruirsi una Unit contenente procedure che emulino la BGI (anche questo è un argomento che affronteremo tra breve; un po' di pazienza...). **In tal modo si potranno portare con facilità programmi dal mondo IBM al mondo Amiga.**

K. Pascal contro Lattice C

Ovviamente sarebbe estremamente riduttivo comparare Kick Pascal ad uno dei vecchi (ed inefficienti) compilatori Pascal di Amiga; sarebbe come se si volesse confrontare la velocità di un interprete con quella di un compilatore.

Viene quindi pubblicato, in queste pagine (ed adattato opportunamente) il programma dimostrativo **Lines.c**, ben noto a chi possiede il compilatore Lattice C 5.04 (è infatti presente tra i file di esempio nella confezione originale).

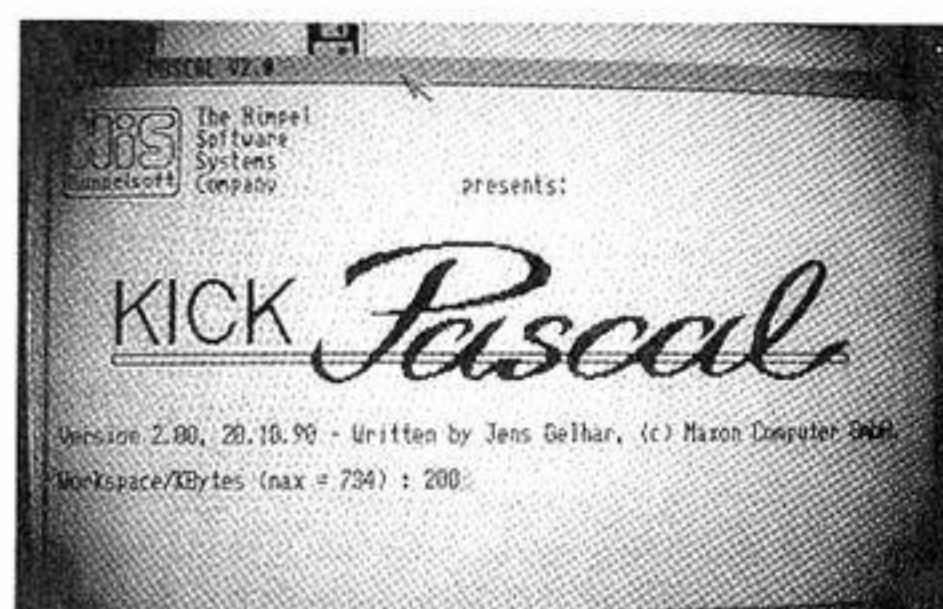
Il lavoro di "traduzione" da "C" a Kick Pascal non ha presentato particolari problemi; ha richiesto, all'incirca, una trentina di minuti, prevalentemente impiegati nel sostituire le istruzioni di assegnazione ("=" del C con ":=" del Pascal) e le definizioni delle variabili. L'interfaccia con le librerie di sistema è, a parte qualche rara eccezione, identica in entrambi i compilatori e non ha presentato alcun problema.

Il codice generato da Kick-Pascal è di qualità paragonabile a quello prodotto dal Lattice C, pur se penalizzato da un codice di startup forse un po' lungo.

Test di compilazione

Sono stati eseguiti alcuni test, tenendo conto (vedi tabella) soprattutto di un dato fondamentale (il tempo di compilazione) variabile molto critica, soprattutto per chi non possiede un Hard Disk.

Bisogna ammettere che i risultati sono sbalorditivi: Kick Pascal, a mano a mano che compila programmi, mantiene in memoria i file **Include** e le **Unit** utilizzate nelle precedenti compilazioni (i pochi che non dispongono di almeno 1 MB di memoria non potranno avvantaggiarsi moltissimo di tale caratteristica).

I risultati di questa tecnica si fanno sentire, pesantemente; dopo la compilazione del primo programma di una sessione di lavoro con Kick Pascal, praticamente scompare il divario di velocità di compilazione tra sistemi con Hard Disk e sistemi con Floppy Disk, croce di moltissimi altri compilatori (leggasi **C** e **Modula-2** per Amiga...).

Ci troviamo, in conclusione, di fronte ad un ottimo compilatore Pascal in ambiente AmigaDos, vicinissimo agli standard Pascal cui sono abituati gli utenti Ms - Dos, con il vantaggio di avere una macchina come Amiga che, se sfruttata a dovere, dovrebbe far apparire l'Ms - Dos un sistema inadeguato e, quantomeno, vecchiotto.

```
program LinesDemo;

{LINESDEMO.p}
{ Programma originale:}
{ LINES.c - a line drawing demo for Amiga}
{by John Riley, Lattice, Inc.}
{ (LATTICE C v5.04)}
{Tradotto in KICK-PASCAL v2.0}
{a scopo dimostrativo}
{da Filippo Bosi - Lugo (RA)}

{per COMPUTER CLUB}

{$incl"graphics.lib"}
{$incl"intuition.lib"}

const MaxLen = 100;

var

wakeup,code: short;
```

```
w : p_window;
rp: p_rastport;
cdrp : Ptr;
message2 : p_IntuiMessage;
x,y,xd,yd: array[0..1] of integer;
co:integer;
class:long;
ox,oy:array [0..1] of array [0..15] of integer;
xx:array [0..127] of integer;
xlim,ylim:integer;
ishort: short;
i,k: integer;
j: long;


Function event:boolean;

begin
 case class of
  _CLOSEWINDOW:
   event:=FALSE;
  NEWSIZE:
   begin
   xlim := w^.Width;
   ylim := w^.Height;
   event:=TRUE;
   end;
  else
  event:=TRUE;
 end;
end;


begin
 w:=Open_Window(100,100,300,100,$0201,
     _CLOSEWINDOW+NEWSIZE,
      WINDOWDEPTH+WINDOWDRAG+WINDOWCLOSE+
      GIMMEZEROZERO+WINDOWSIZING,
      'Nervous Lines (KICKPASCAL version)',
       Nil,
       50,50,640,400);

 OpenLib(GfxBase,"graphics.library",0);
 if(GfxBase = Nil) then Halt(10);
 OpenLib(IntBase,"intuition.library",0);
 if(IntBase = Nil) then
   begin
   CloseLib(GfxBase);
   Halt(10);
   end;

 rp := w^.RPort;    { Get the raster port pointer }

 SetAPen(rp,3);    { Set foreground pen to white }
 SetDrMd(rp,JAM1);   { Draw with foreground pen }

 xlim := w^.Width;
 ylim := w^.Height;
 for i:=0 to 1 do
 begin
  x[i] := random(xlim) + 1;
  y[i] := random(ylim) + 1;
  xd[i] := random(10) + 1;
  yd[i] := random(10) + 1;
end;

for i:=0 to 1 do for j:=0 to 15 do begin
                          ox[i][j]:=0;
                          oy[i][j]:=0;
                                   end;
co := 1;
j := 0;


repeat

 SetAPen(rp,0);
 Move(rp,ox[0][j mod 16],oy[0][j mod 16]);
 Draw(rp,ox[1][j mod 16],oy[1][j mod 16]);
 SetAPen(rp,co);
 Move(rp,x[0],y[0]);
 Draw(rp,x[1],y[1]);

if(random(10) < 2) then
   begin
    co:=co + 1;
    if (co > 7) then co := 1;
    SetAPen(rp,co);
   end;

 for i:=0 to 1 do
  begin
   ox[i][(j+10) mod 16] := x[i];
   oy[i][(j+10) mod 16] := y[i];
   x[i] := x[i]+xd[i];
   y[i] := y[i]+yd[i];
   if(x[i] < 2) then
   begin
    x[i] := 2;
    xd[i] := -xd[i];
   end
  else if(x[i] > xlim) then
   begin
    x[i] := xlim;
    xd[i] := -xd[i];
   end;
  if(y[i] < 2) then
   begin
    y[i] := 2;
    yd[i] := -yd[i];
   end
   else if(y[i] > ylim) then
    begin
    y[i] := ylim;
    yd[i] := -yd[i];
   end;
  if (random(10) < 2) then
   begin
    if(xd[i] < 1)  then k := 1;
    xd[i] := random(8);
    if(k = 1) then xd[i] := -xd[i];
    k := 0;
   end;

  if(random(100) < 50) then
   begin
```

```
    if(yd[i] < 1) then k := 1;
    yd[i] := random(8);
    if(k = 1) then yd[i] := -yd[i];
    k := 0;
   end;
 end;

j:=j+1;
  if(w^.UserPort^.mp_SigBit <> 0) then
   begin
   message2 := Get_Msg(w^.UserPort);
    if(message2 <> Nil) then
```

```
       begin
       class := message2^.Class;
       code := message2^.Code;
       Reply_Msg(message2);
       end;
    end;
 until (not (event));

 CloseWindow(w);
 CloseLib(GfxBase);
 CloseLib(IntBase);
end.
```

```
Program QCDemo;

{ QCDEMO.p}
{ dimostrativo per KICK PASCAL}
{by Filippo Bosi}
{richiede il file QCDISEGNA.p}

Uses QCDisegna;

var i:long;

begin
```

```
 ApriFinestra;
 repeat
    for i:=1 to 1000 do
      begin
      Cerchio(ACasoX,ACasoY);
      Quadrato(ACasoX,ACasoY);
      end;
    PulisciFinestra;
 until (GadgetChiusura);

 ChiudiFinestra;
end.
```

```
Unit QCDisegna;

{QCDISEGNA.p}
{Unit  per il programma qcdemo.p}

Interface
{queste sono le funzioni}
{messe a disposizione di un}
{ programma Kick-Pascal che}
{specifichi "USES Disegna;"}

Procedure ApriFinestra;
Procedure Cerchio(x,y:long);
Procedure Quadrato(x,y:long);
Procedure ChiudiFinestra;
Function  GadgetChiusura:boolean;
Function  ACasoX:long;
Function  ACasoY:long;
Procedure PulisciFinestra;

Implementation

{tutto cio' che sta in questa sezione}
{non e' visibile al programma}
{che utilizza la unit, a meno che}
{non sia stato incluso nella}
{sezione INTERFACE}

{$incl "graphics.lib"}
{$incl "intuition.lib"}
```

```
Const MaxColor = 7;
{ numero ma. colori schermo WorkBench}
      Altezza = 100;
      Larghezza = 400;
{ -- dimensioni della finestra -- }

Titolo = 'Quadrati&Cerchi demo by F.Bosi';

var rp:p_RastPort;
    Win:p_Window;
    xlim,ylim: long;


Procedure ApriFinestra;
{Apre finestra e le librerie intuition e graphics}
begin
 Openlib(IntBase,"intuition.library",0);
 Openlib(GfxBase,"graphics.library",0);

  Win:=Open_Window(80,40,Larghezza,Altezza,$0100,
_CLOSEWINDOW,
WINDOWCLOSE,  Titolo,  Nil,  Larghezza,  Altezza,
Larghezza, Altezza);
 if Win = Nil then begin
Writeln('Unit DISEGNA - Procedura ApriFinestra:');
 Writeln('Non riesco ad aprire la finestra !!');
 Halt(20);
 { torna al s.o. con codice di errore 20}
                  end;
 rp:=Win^.RPort;
```

```
 xlim:=Win^.Width;
 ylim:=Win^.Height;
end;


Procedure PulisciFinestra;
{ -- cancella il contenuto della finestra -- }

begin
 ClearScreen(rp);
 RefreshWindowFrame(Win);
end;


Function ACasoX;

{ -- ritorna un numero a caso entro il limite massimo
x della finestra -- }

begin
ACasoX:=Random(xlim);
end;


Function ACasoY;
{ -- ritorna un numero a caso entro il limite massimo
y della finestra -- }

begin
ACasoY:=Random(ylim);
end;


Function GadgetChiusura;
{ -- ritorna TRUE se e' stato premuto il gadget di
chiusura finestra -- }

var message2 : p_IntuiMessage;
    class,code: long;

begin
   GadgetChiusura:=FALSE;

   if(Win^.UserPort^.mp_SigBit <> 0) then
    begin
    message2 := Get_Msg(Win^.UserPort);
     if(message2 <> Nil) then
      begin
      class := message2^.Class;
      code:= message2^.Code;
      Reply_Msg(message2);
      end;
             if  class  =  _CLOSEWINDOW  then
GadgetChiusura:=TRUE;
    end;
end;


Function NeiLimiti(x,y,r:long):boolean;
{ Ritorna TRUE se una figura sta in finestra }
var maxx,maxy,minx,miny: long;

begin
 maxx:=x+r+4;
 minx:=x-r-4;
 maxy:=y+r+4;
 miny:=y-r-10;
 NeiLimiti:=(maxx < xlim) and (minx > 0) and (maxy
< ylim) and (miny > 0);
end;

Procedure Cerchio;
{ -- disegna una circonferenza di raggio casuale
solo se sta nella finestra -- }

var r:long;

begin
 r:=random(80);
 if NeiLimiti(x,y,r) then begin
             SetAPen(rp,random(MaxColor));
           DrawEllipse(rp,x,y,r,r);
           end;
end;


Function Max(a,b:long):long;
{ -- ritorna il valore massimo tra a e b -- }

begin
 if a>b then b:=a;
 Max:=b;
end;


Procedure Quadrato;
{disegna un rettangolo solo se sta nella finestra}

var r1,r2:long;

begin
 r1:=random(40);
 r2:=random(40);
 if NeiLimiti(x,y,Max(r1,r2)) then begin
          SetAPen(rp,random(MaxColor));
          RectFill(rp,x,y,x+r1,y+r2);
          end;
end;

Procedure ChiudiFinestra;
{chiude la finestra e le librerie intuition e
graphics}

begin
 Close_Window(Win);
 Closelib(IntBase);
 Closelib(GfxBase);
end;

{codice di startup della Unit}
{viene eseguito una sola volta all'inizio del
programma }

begin
Randomize;
end.
```

A cura di Luigi Callegari

AmiGazzetta N. 11, un dischetto per "Amigo"

AmiGazzetta è un dischetto venduto come supplemento di Commodore Computer Club, contenente materiale originale tratto dalla Rivista o di pubblico dominio, selezionato tra quanto di meglio esistente nell'abbondante circuito internazionale per Amiga

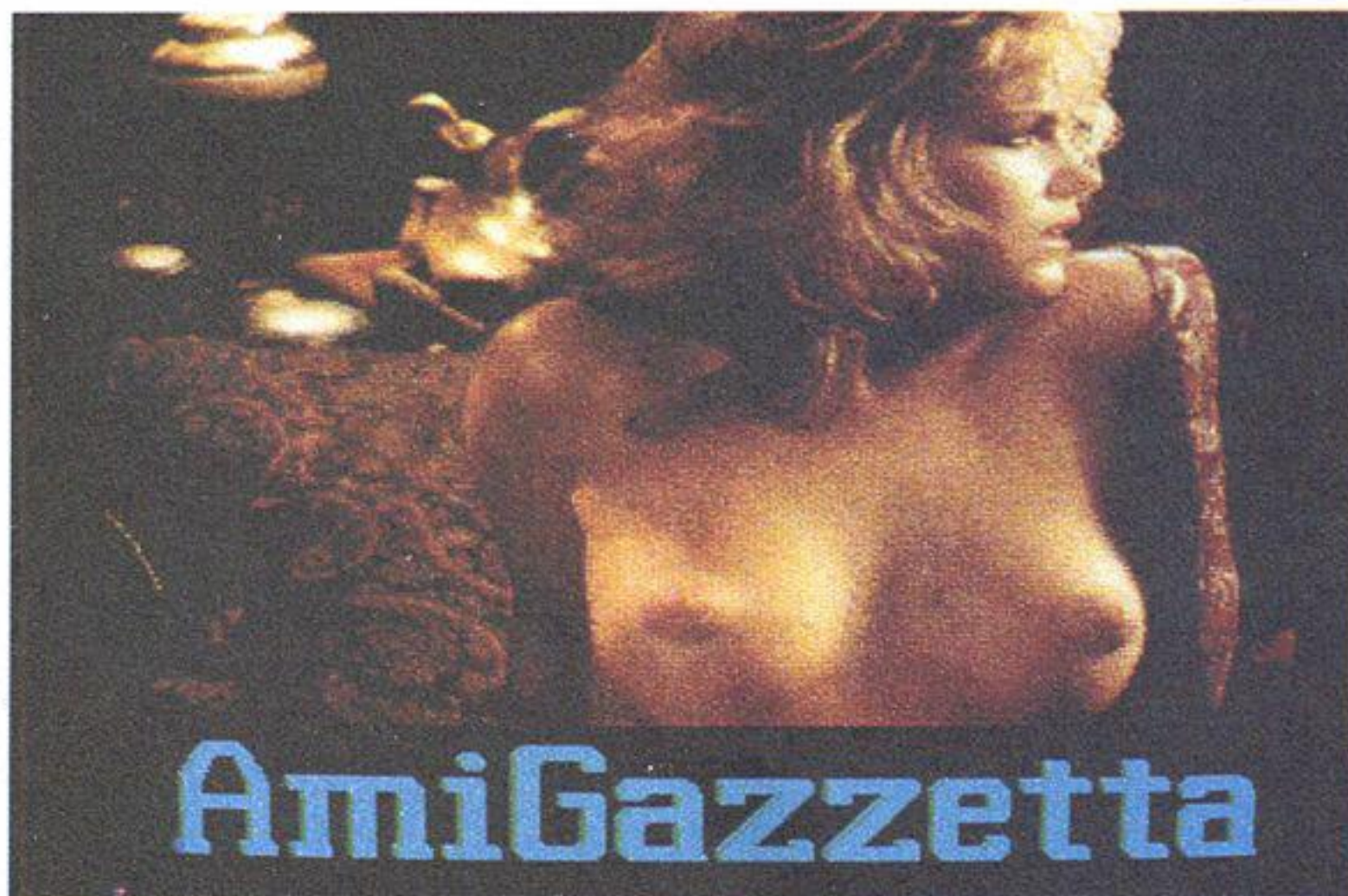
AmiGazzetta è un dischetto venduto come supplemento di **Computer Club**, contenente materiale originale tratto dalla Rivista o di pubblico dominio, selezionato tra quanto di meglio esistente nell'abbondante circuito internazionale per Amiga.

Il dischetto consiste di una raccolta di file eseguibili, in gran parte sia da Workbench che da CLI, tutti accompagnati dalla originale documentazione in file ASCII separati, leggibili (e stampabili opzionalmente su carta) da Workbench o CLI con una apposita utility fornita. Tutti i file sono verificati per funzionare con qualunque tipo di hardware (ECS, Amiga 500, 1000 e 2000) e software (Kickstart V1.2/1.3), scelti tra quelli presenti su numerosi dischetti di pubblico dominio, selezionati tra le ultime novità.

Ai programmi veri e propri possono accompagnarsi saltuariamente **icone** da usare con i propri programmi, file di **documentazione** supplementari (come l'elenco delle banche dati ad accesso **gratuito** della rete Fidonet di questo numero, oppure l'elenco di tutti gli articoli apparsi su Amiga nella pagine di **C.C.**), dimostrativi grafici e tutto ciò che di altro può interessare degli *utenti attivi* di Amiga.

Tutti i file, sia gli eseguibili sia le documentazioni, sono sottoposti a compressione con un apposito programma (**PowerPacker** Professional V3.0a) in modo da stipare quanto più materiale possibile sul dischetto.

I file eseguibili vengono riportati alla condizione originale ("scompattati") ed eseguiti come di consueto in modo del tutto trasparente all'utente, così come i file di documentazione che vengono decompressi e visualizzati (o stampati) dall'apposito programma **PPMore**.

In questo modo, mantenendo praticamente inalterata la rapidità ed immediatezza d'uso e di accesso dei file sul disco, si può materialmente stipare su ogni numero di Amigazzatte una quantità di dati del **30 - 40% superiore** agli 880 K standard.

Procuriamoci Amigazzetta

Il dischetto AmigaGazzetta N. 11, in formato AmigaDOS, può essere acquistato **soltanto** per corrispondenza, inviando la misera somma di lire 12.000 (dodicimila) a mezzo conto corrente postale numero **37952207** intestato alla **Systems Editoriale**.

Si rammenti che non è possibile richiedere la spedizione per contrassegno e che è importante specificare il proprio indirizzo completo ed il numero di AmiGazzetta desiderato.

Collaborazione

Chi volesse collaborare inviando il proprio materiale a AmiGazzetta, deve preparare un dischetto da 3,5" in formato AmigaDOS contenente il file eseguibile, un file di documentazione (anche sommario) in formato ASCII puro (senza codici di controllo nè di impaginazione, scritto con un normale editor ASCII o con un programma di videoscrittura e salvato in formato ASCII piano) che illustri l'uso pratico del programma e, eventualmente, il file sorgente. E' importante inviare anche una autorizzazione scritta e firmata, su carta e non in un file, dove si dichiara di sottoporre all'attenzione della Systems Editoriale il proprio software (citandolo per nome), garantendo che si tratta di lavoro personale ed originale non comprendente parti soggette a copyright, autorizzando l'editore alla pubblicazione, nei tempi e nei modi da esso voluti, senza richiedere alcun compenso e senza fine di lucro da parte di alcuno, come materiale di pubblico dominio.

Oltre ai programmi, in qualunque linguaggio per Amiga (Amos, Assembler, AmigaBasic, C, Fortran, Pascal, Assembler, Modula-2, Draco, Lisp ecc. ecc.), si possono naturalmente inviare proprie elaborazioni grafiche, icone e traduzioni di programmi di pubblico dominio, sempre che tali traduzioni non violino quanto sancito dall'autore del programma.

Contenuti

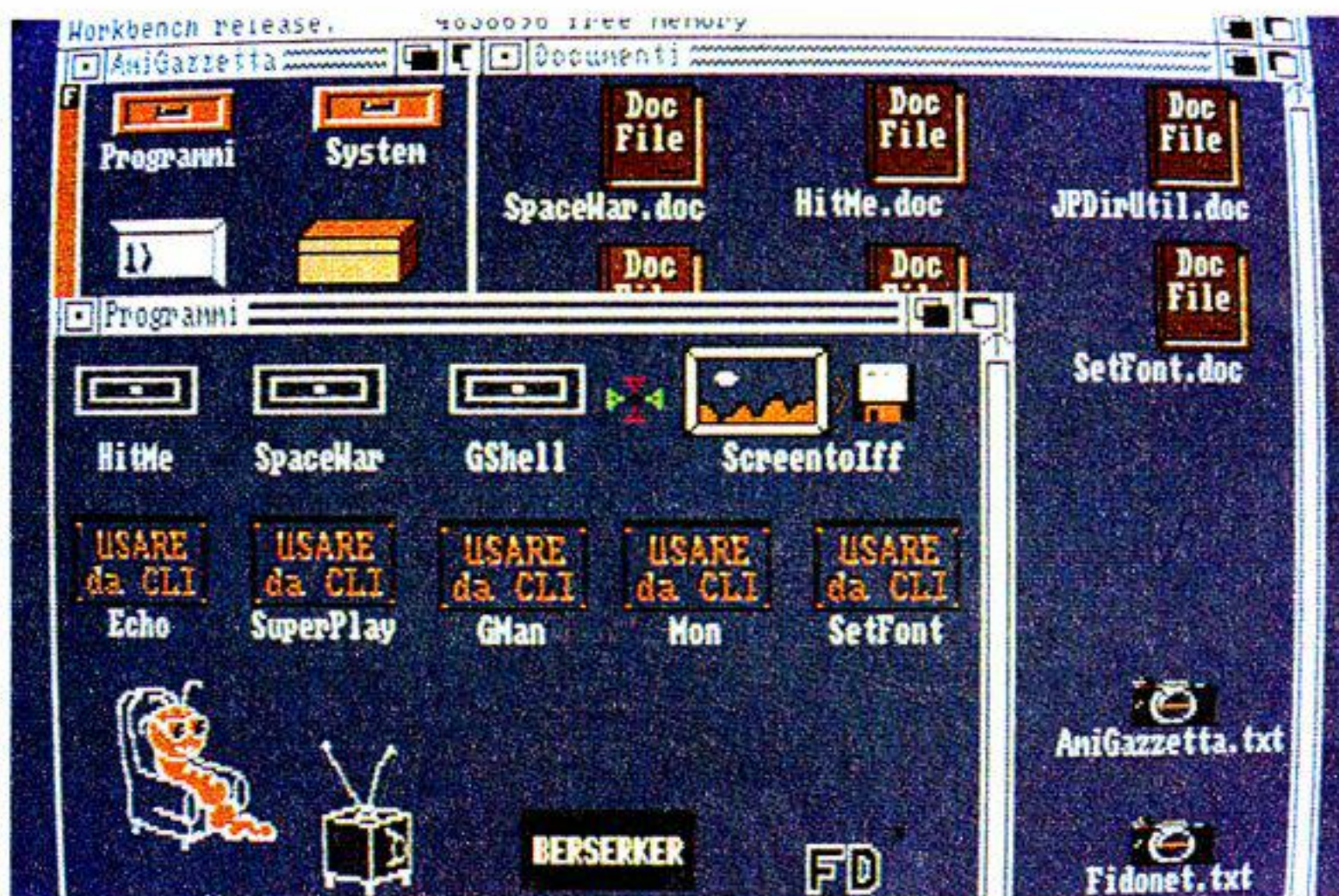
Gli appassionati di grafica non potranno che apprezzare il famoso programma **Mostra**, scritto da *Sebastiano Vigna* (Sviluppatore Commodore milanese, ben noto nel circuito di pubblico dominio internazionale).

Si tratta probabilmente del più completo **visualizzatore di file grafici IFF**, in grado di visualizzare qualunque formato di file in ogni modo grafico ed a qualunque risoluzione (comprese le cosiddette "intermedie"). Le istruzioni (complete, in lingua originale inglese) sono naturalmente incluse nel dischetto. **Chi acquista il dischetto Amigazzetta 11 riceverà anche la completa traduzione in italiano.**

Molto più ludico il programma **HitMe**, che esegue un test della velocità dei riflessi e di spostamento del mouse dell'utente, con tanto di temporizzazioni in decimi di secondo, effetti sonori e classifica su disco.

GMan è invece un utilissimo programma per ottimizzare i contenuti dei propri dischi, particolarmente godibile da chi possiede **Hard Disk**. Infatti tale programma può reperire e segnalare il nome (completo di path) di tutti i file "doppi" in un dato volume o serie di directory.

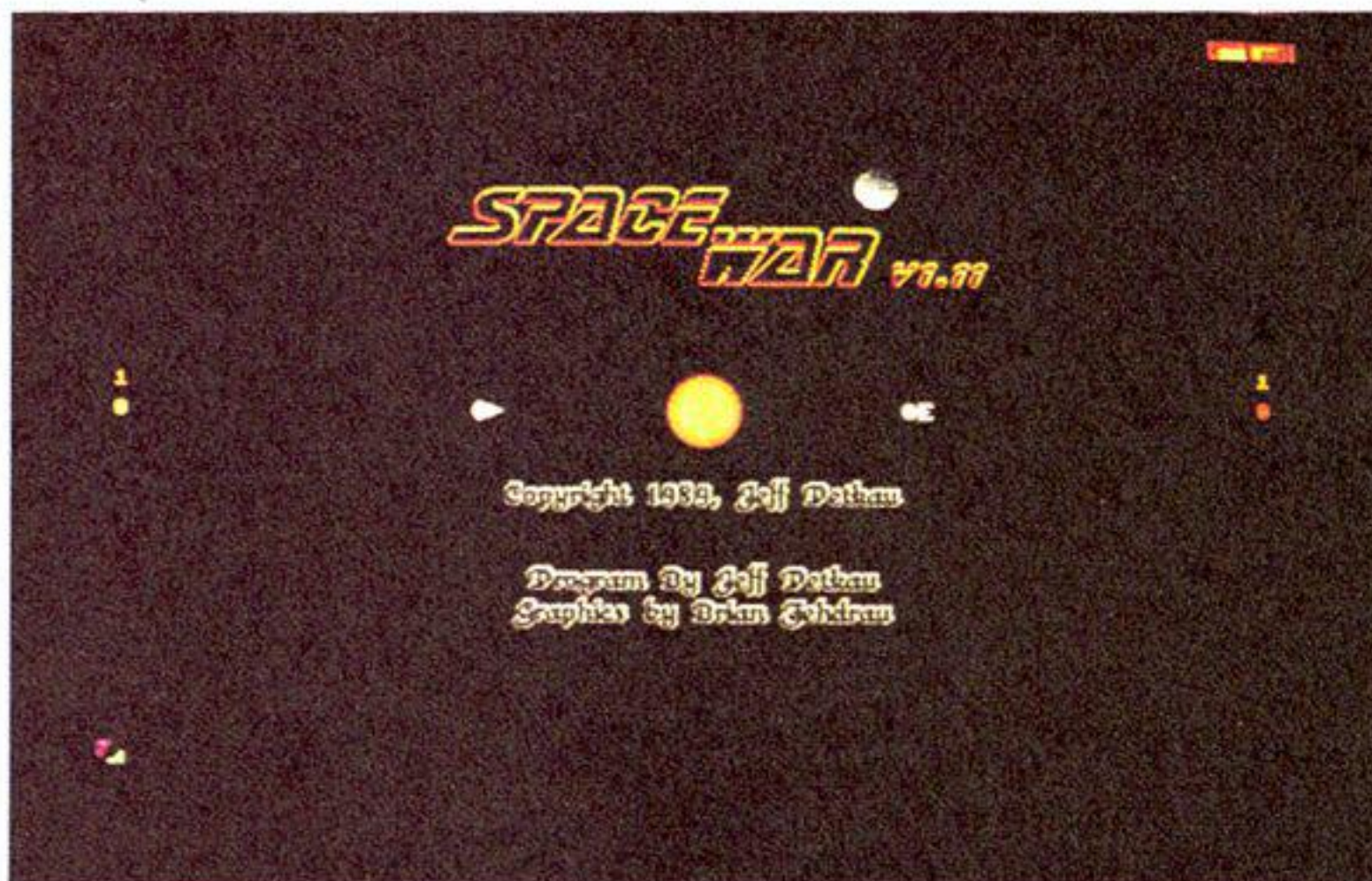
Altro programma serissimo è **MON** di Timo Rossi. Si tratta di un completo monitor, in grado di eseguire circa **40 funzioni**, tra le quali lo spostamento di **aree di memoria**, l'esecuzione **passo passo** o con **breakpoint** di un programma as-

sembly, la gestione degli **hunk** di un file eseguibile, la gestione delle risorse di **Exec**, lo I/O a basso livello da disco e molto altro ancora.

Anche in questo caso, **le istruzioni complete tradotte in italiano verranno inviate a chi acquista il disco** Amigazzetta 11.

SetFont (V2.7) è l'ultima versione di una utility scritta da Dave Haynie (uno dei padri dell'Hardware dell'Amiga 3000) in grado di sostituire la consueta fonte "**Topaz 8**" del Workbench con una qualunque altra, consentendo nel contempo una grande varietà di controlli.

SuperPlay è un riproduttore di file sonori in formato IFF 8SVX od altro. Sono previste varie opzioni per controllare il volume, la velocità di riproduzione, l'esecuzione da file batch di più motivi in una directory ed altro ancora.

Il nome non deve ingannare, **Echo** è una versione speciale dell'omonimo comando Shell che consente una grande varietà di opzioni, tra cui la pulizia dello schermo, la collocazione del testo ad una qualunque posizione della finestra di output, lo scrolling verso l'alto o verso il basso e la generazione semplificata di codici ANSI di controllo.

ScreenTolff è un programma che rimane "residente", ovvero gira in sottofondo agli altri task ed in modo invisibile, permettendo in qualunque momento alla pressione di un tasto di fare comparire un requester e di salvare lo schermo attualmente visualizzato come file IFF.

Berserker IVa è l'ultima versione di un antivirus scritto interamente in linguaggio macchina da Ralf Thanner. E' in grado di riconoscere istantaneamente molti tipi di virus da bootblock e link ed è studiato in modo da riconoscere, tramite severo controllo di vitali parametri, la presenza di qualunque tipo di virus in memoria, anche attualmente non conosciuto.

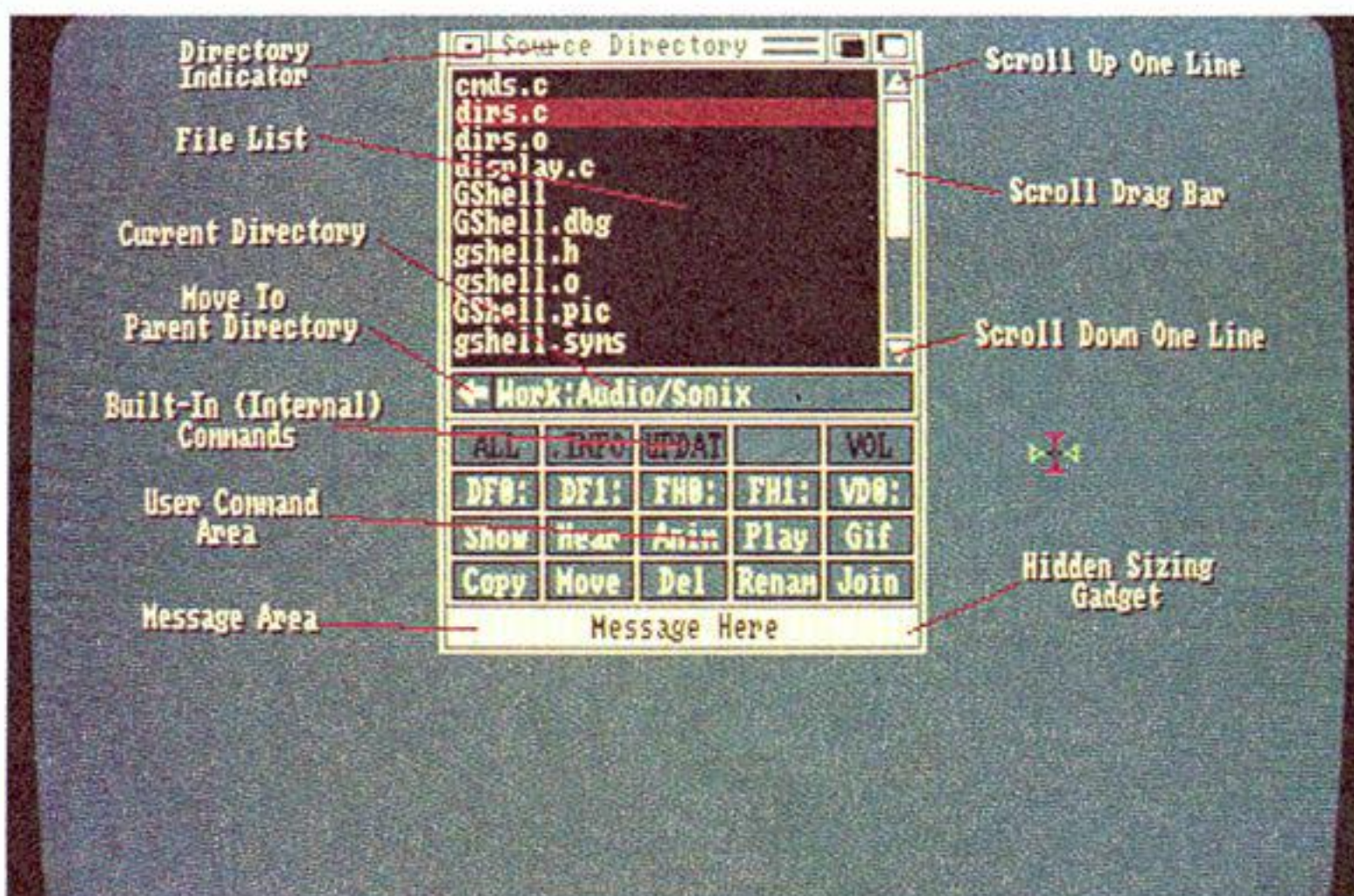
GShell è una semplice e compattissima utility per la **manipolazione di file** e directory, che consente cioè di eseguire da Workbench (con mouse, gadget e menu) operazioni normalmente eseguite a suon di polpastrelli da Shell.

FlashDisk è un ottimizzatore di dischetti, che consente di copiare un floppy su di un altro (occorrono due drive) ridisponendone la struttura DOS in modo che successivi accessi da Workbench o da CLI risultino molto più veloci. Controllato da gadget, prevede tre modi di ottimizzazione (deframmentazione dei file, sistemazione per accesso rapido da Workbench e misto) ed altre opzioni.

SpaceWar è un semplice giochino per due avversari, il cui scopo è manovrare da tastiera o joystick la propria astronave per evitare di collidere con pianeti, manovrando in modo utile, a dispetto delle forze gravitazionali, per abbattere l'avversario.

Il programma "Mostra"

Vi sono in circolazione parecchie utilities per visualizzare i files IFF ILBM, ma nessuna è veramente completa: se un visualizzatore può mostrare disegni in overscan, ciascun grafico con più di 640 pixel su di una linea viene shiftato come se fosse a 700 pixel; alcuni programmi possono visualizzare soltanto una directory di grafici costringendo l'utente ad immettere tutti i nomi dei files; altri vanno in crash se si passano loro grafici in PAL, altri ancora sporcano il grafico con la barra di intestazione se si preme il pulsante destro del mouse.

E' raro disporre di un'opzione che forzi un modo schermo (per i tecnici: i vecchi files IFF ILBM non avevano il chunck CAMG!) e di un file requester come si

deve quando si ignora il nome del file (chi si ricorda i nomi dei files di Grabbit?).

Il programma **Mostra**, che elimina questi (ed altri) problemi, fa un utilizzo massiccio della libreria **ARP**. Il pattern matching, il tracking delle risorse, le funzioni residenti e tutto il resto sono utilissime, con l'unico svantaggio di dovere inserire il file **ARP.library** nella directory associata al device logico **LIBS:**. In caso contrario Mostra non funzionerà.

Mostra può essere invocato dal Workbench: chiamato da solo presenta il requester di file e mostra il grafico prescelto sinchè non si sceglie Cancel o non si chiude la finestra del requester Arp. Altrimenti si possono scegliere più icone, tramite il meccanismo della selezione multipla, oppure si può modificare appropriatamente il Default Tool delle icone associate ai grafici in modo che lo richiamino. Facendolo con un **Cat** o **List** si ottiene uno *Slideshow Automatico*.

Mostra può trattare **qualunque** tipo di file IFF ILBM (inclusi files di **Photon Paint**, **Excellence!** eccetera). Questo significa che prevede non solo i **FORM IFF ILBM**, ma anche gli **ILBM** nidificati, **FTXT**, **FORM**, **CAT**, **LIST** e **PROP**! Tenta sempre di trovare il tipo di schermo idoneo e non si ribella se si chiede di vedere brush di 1x1 in grafici molto grandi (sino a **5120 x 4096**). In ogni modo è consentito forzare il tipo di schermo con alcune opzioni.

Una delle caratteristiche principali in questa versione è il supporto diretto dello **SHAM** ed il supporto indiretto dei modi Dynamic HAM e Dynamic Hi-Res. Si noti che si ha un completo supporto dello Sham e che si possono fare scorrere grafici in questo formato in schermi piccoli.

Ovviamente i movimenti verticali richiedono parecchio lavoro da parte di Amiga. Per il Dynamic Hi-Res ed il Dynamic HAM il programma Mostra chiama il programma **Dyna-Show** della NewTek, che deve essere presente in una delle path di ricerca comandi del vostro sistema (il modo più semplice è di porlo in **C:**). In questo modo si può vedere qualunque tipo di immagine, ma ovviamente nel modo Dynamic si perdono alcune delle caratteristiche più potenti di Mostra, come ad esempio il dimensionamento dello schermo.

Ad esempio, la seguente linea Shell...

```
M Mypics:* SECS 5 FADE 1 NOMOUSE
BLACKBACKGROUND REPEAT CENTER ALL
```

...genera un vero, infinito, slideshow dei grafici nella directory `MyPics:` ed in tutte le sue subdirectory, centrando e senza puntatore del mouse. Per sospendere lo slideshow si deve usare **Ctrl-C**.

Ancora...

```
M df0:*.image df0:Pictures/*.pic
HIRES LACE
```

...mostra tutti i files che terminano con **.image** collocati nel drive **DF0:** ed i files che terminano con **.pic** nella directory **df0:Pictures**. Mostra forzerà l'uso dell'alta risoluzione, con schermi interlacciati. Attivando il flag **HIRES** sui grafici con più di 4 bitplanes solitamente porta a non vedere alcunchè.

```
M dh0:hirespic LORES NOLACE
```

...mostrera' un grafico in alta risoluzione in bassa risoluzione ("zoomandoci"). Ci si può spostare completamente con i tasti cursore come descritto sopra.

Caratteristiche avanzate

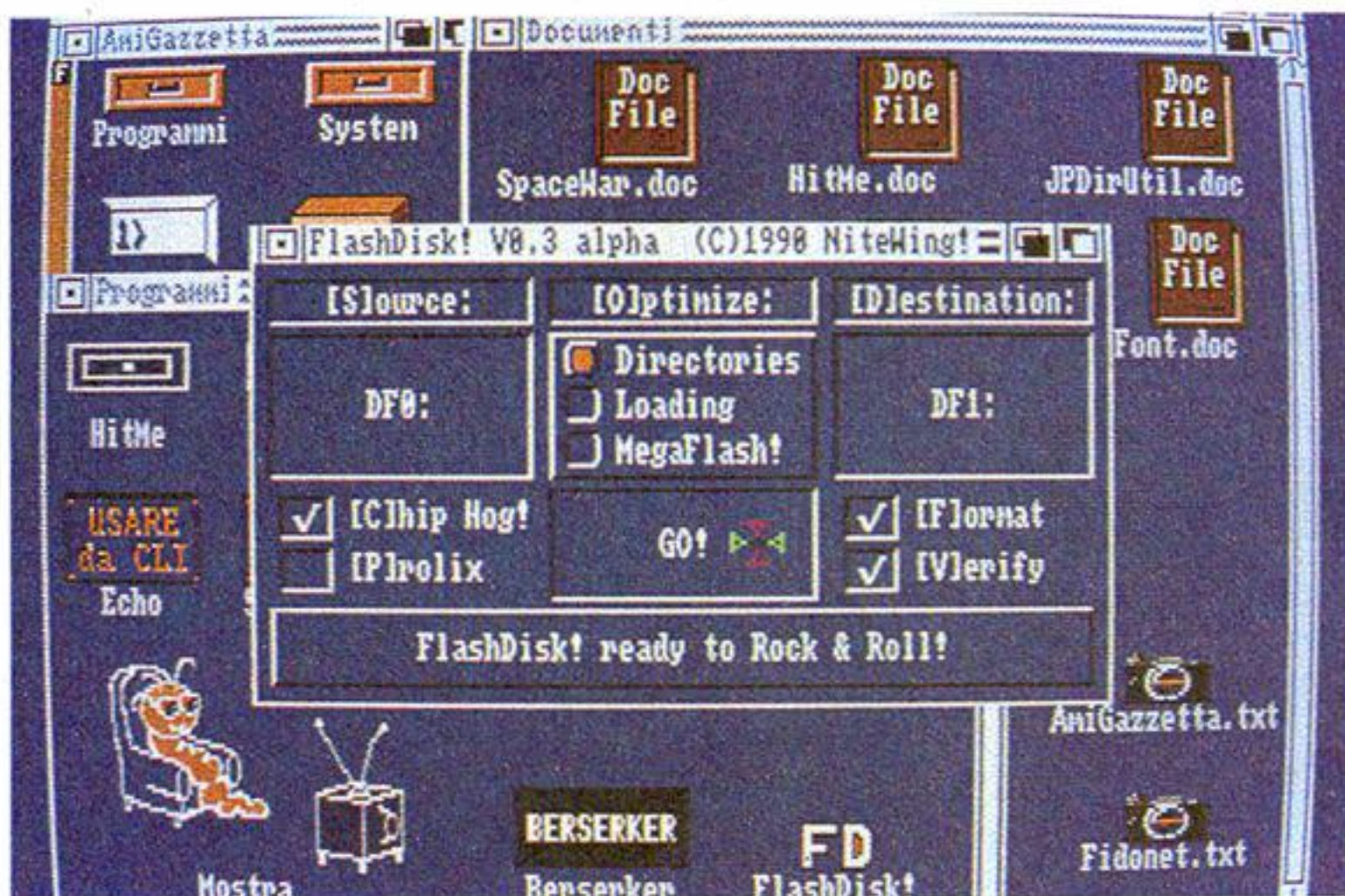
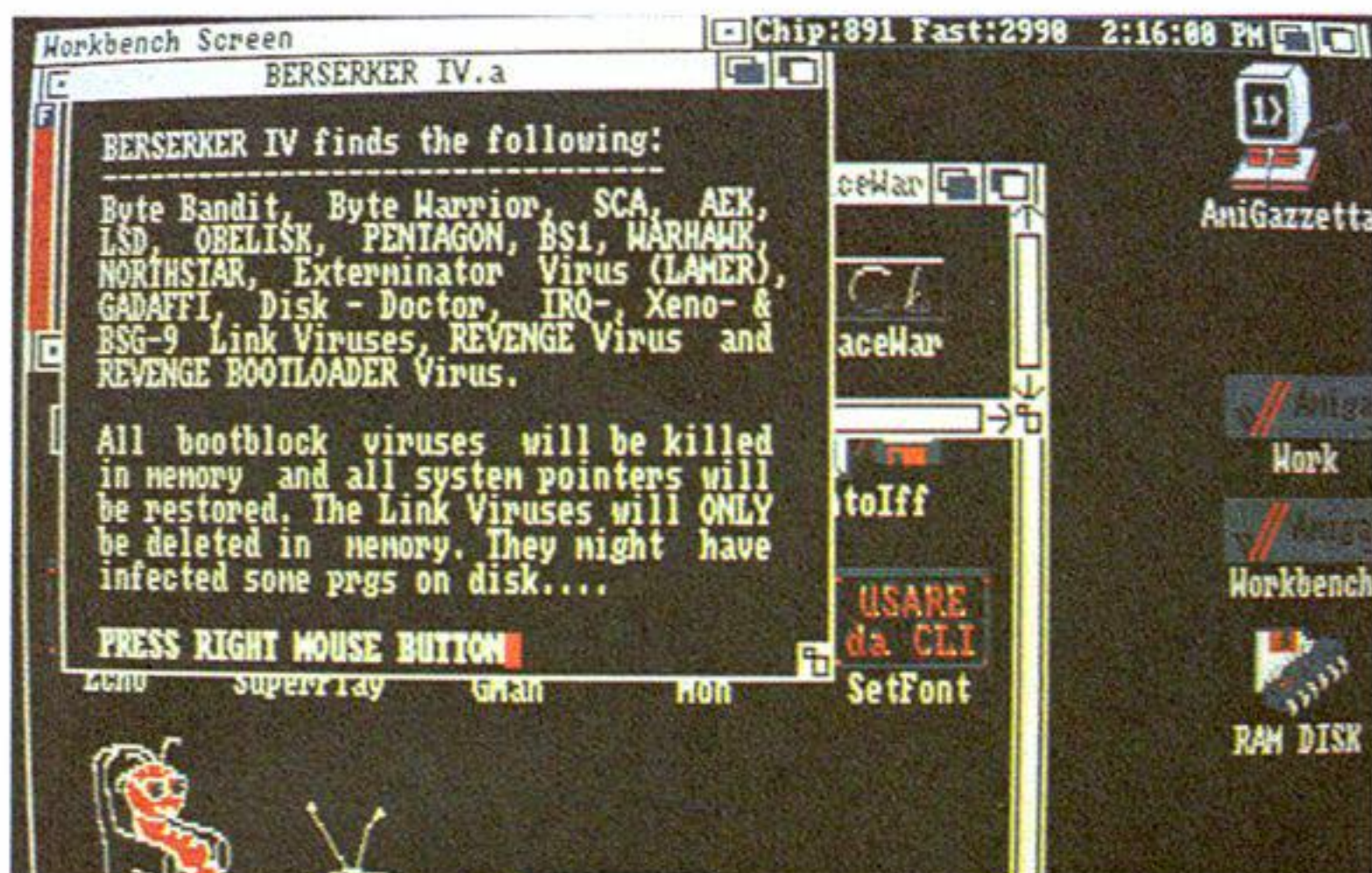
Per la massima flessibilità, Mostra consente di usare files di startup e ToolTypes Startup, con i quali si configurano i programmi per ogni esigenza.

Il file di startup standard si chiama **Startup-Mostra** e deve essere contenuto in **S:**. Mostra lo cerca quando viene avviato dal CLI. Il formato di questo file di avviamento è esattamente lo stesso del formato di linea di chiamata del CLI, tranne, ovviamente, il nome del comando "M". I comandi possono essere spezzati in più linee o raggruppati in una singola linea. Ciascun switch o keyword nel file di startup agisce come default e viene sovrascritto o commutato da qualunque argomento della linea di comando.

Gli switch della linea di comando agiscono come commutatori (se il file di startup ha la parola ALL e si usa "M * ALL" NON si andra' nelle subdirectory), mentre le keyword agiscono semplicemente in base al valore (usando Fade 1 nel file di startup e dicendo Fade 3 nella linea di comando i grafici vengono sfumati a velocità 3).

Le opzioni concernenti i files di startup sono **Nostartup** e **Batch**. La prima inibisce la ricerca dei files di startup, la seconda vuole un completo path con nome del file che Mostra usa come file di startup.

Si possono fare le stesse cose con una icona che non ha il file collegato e porre nel ToolType delle wildcards; le wildcards saranno poi mostrate con quelle opzioni. Non usando invece le wildcards si ottiene un'icona di stile: usando Shift e cliccando su alcune icone, Shift e cliccando su di una icona di stile (in quest'ordine!) e poi Shift e doppio click sull'icona Mostra (o direttamente Shift e doppio click sull'ultima icona, se il suo default tool è Mostra): si vedranno i files grafici selezionati con le opzioni specificate nell'icona di stile. Questa richiama da sè il requester di file. Si noti che anche un'ico-

na con wildcards può essere usata come icona di stile dal momento che le selezioni multiple prendono il posto delle wildcards selezionate.

Ecco un esempio di problemi comuni e possibili soluzioni.

"Ho centinaia di grafici sul mio Hard Disk e le ho organizzate in vari gruppi. Come posso fare dei semplici slideshow con ciascun gruppo separato?"

Semplice: si crea un'icona di progetto per ciascun gruppo e si aggiunge un ToolType Startup, poi si aggiunge un ToolType come `PICS:Group1/* ALL SECS 5 CENTER` per scegliere queste opzioni, si fissa il tool di default a "C:M" e si clicca due volte (ovviamente, questo deve essere fatto per ciascun gruppo di icone). Non si diano alle icone gli stessi nomi di una directory, o Mostra andrà in confusione.

"Voglio sempre vedere lo schermo nero quando parto da WB".

Si metta nel Tooltype di Mostra **Blackbackground.**

"Ho diversi grafici. Talvolta vorrei vederne alcuni a schermo completo e senza mouse, altre volte in schermo 128 x 128, altre volte centrato".

Si creino tre icone di stile. Ciascuna icona deve avere come primo ToolType Startup e le seguenti devono essere qualcosa come Nomouse, Width 128 Height 128 e Center. Il tool di default deve essere C:M. Quando si desidera vedere in un certo modo, si clicca con Shift le icone del grafico e poi si fa un doppio click con Shift dell'icona dello stile voluto.

"Voglio cliccare i miei grafici uno per volta, ciascuno con opzioni differenti e mi piacerebbe anche vederli in gruppi irregolari, ciascuno con le sue opzioni"

Occorre uno psicanalista, non Mostra!

Scherzi a parte, il programma Mostra è realmente potente e consente la gestione sofisticata di qualsiasi tipo di grafico. Ripetiamo che il manuale completo (in italiano) viene inviato solo a chi acquista il N. 11 di Amigazzetta.

AmigaMon V1.24

E' stato scritto da Timo Rossi (Kellankoski 44300 Konnevesi - Finland)

E' un programma di **monitor/debugger** per l'Amiga molto sofisticato, ispirato a vari monitor del **Commodore 64**. E un programma rientrante (puro) che può essere reso residente col comando Resident del Workbench V1.3 o **ARes** della ARP.

E previsto l'editing della linea usando i tasti sinistro e destro di spostamento del cursore, il backspace per cancellare il carattere prima del cursore e Del per cancellare il carattere sotto il cursore. Si possono poi usare i tasti cursore per recuperare le linee di comando precedenti (il Monitor *ricorda* le ultime dieci). Shift CursorUp legge l'ultimo comando e lo esegue automaticamente senza premere il Return. Il modo Assembler si può premere Ctrl- E per redigere le istruzioni assembly correntemente correntemente memorizzate a quell'indirizzo.

Per sospendere l'output si può usare, in molti casi, la barra spaziatrice e ripetere la pressione del tasto per riprendere. Per fermare permanentemente l'output si usa, come al solito, Ctrl C.

Il monitor alloca attualmente 2K di spazio per il proprio stack, il rimanente è a disposizione per il programma che si sta "debuggando". Ovviamente la dimensione dello stack usato è modificabile col comando Stack dello Shell.

La base numerica di default è esadecimale, ma la si può cambiare con il comando **ba** (l'argomento del comando è sempre in decimale). I prefissi usati per identificare le basi sono $ (esadecimale); @ (Ottale); % (Binario); _ (Decimale).

I numeri possono anche essere immessi come stringhe ASCII, ad esempio "**FORM**" si può immettere come la stringa esadecimale "464f524d".

L'attuale versione di AmigaMon consente di usare spesso espressioni numeriche al posto di semplici numeri. Il comando di calcolo "**?**" consente di visualizzare direttamente valori come espressioni. Sono disponibili i seguenti operatori: +, -, *, /, %% (modulo), (shift sinistro), (shift destro). Si possono usare parentesi per raggruppare operazioni nelle espressioni. Il simbolo "*" rappresenta anche l'indirizzo corrente, mentre [nomereg] rappresenta il valore del registro "nomereg". Tutti i calcoli sono eseguiti in artimetica intera a 32 bit, senza controllo dell'overflow.

Oltre ai numeri ed alle funzioni si possono usare anche le variabili in espressioni, purchè precedentemente definite. Le variabili sono definite col comando **Set** e sono interi a 32 bit. I nomi di variabili possono contenere qualunque carattere alfanumerico, ma non iniziare con un numero. Non viene tenuto conto delle differenti dimensioni dei caratteri nei nomi.

Le stringhe vengono usate in alcuni comandi e nelle direttive dell'assemblatore. Sono serie di bytes rappresentate da caratteri ASCII tra singoli apici.

Anche **le istruzioni del programma AmigaMon (in italiano) vengono inviate** solo a chi acquista il N. 11 di Amigazzetta.

di Domenico Pavone

A. Replay 2, Genlock mk2; solo per Amiga 500 e 2000

Ad un anno di distanza, parliamo nuovamente di un presidio ormai largamente acquisito tra gli utenti Amiga: la cartuccia Action Replay, ora divenuta Action Replay 2 (AR2)

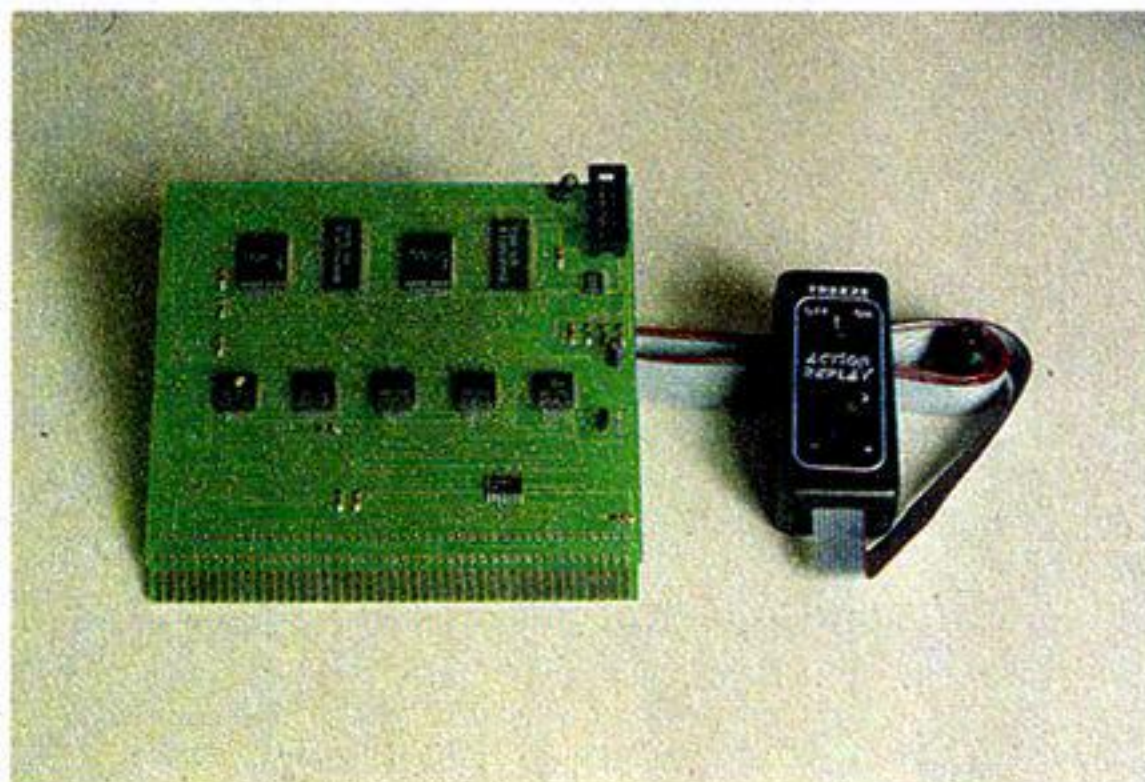
Parlare solo di "cartridge", per ciò che riguarda Action Replay (**AR**), è divenuto limitativo, in quanto ne è commercializzata **anche** una versione su scheda per Amiga 2000, alla quale tra l'altro si farà riferimento in queste pagine (si vedano le foto).

La cartuccia per **Amiga 500**, infatti, esteriormente non si discosta per nulla dal primo modello, e d'altro canto, sotto l'aspetto delle prestazioni, scheda e cartridge risultano assolutamente **identiche**. Se si esclude la descrizione hardware, quanto vedremo varrà dunque per entrambi i modelli.

La differenza immediatamente palpabile tra la "vecchia" Action Replay e la nuova versione, che poi è anche la differenza più netta, consiste nell'**assenza di software** su floppy a corredo. Il motivo è presto detto: il software è stato trasferito su Rom, per cui le funzioni per affrancare i file prodotti con Action Replay dalla presenza fisica della cartuccia, ora non necessitano più di alcun intervento supplementare. Ma prima di approfondire ulteriormente questo aspetto, diamo una breve occhiata all'hardware dedicato ad Amiga 2000.

Si tratta, come si può notare dalle foto, di una scheda per così dire "corta", da inserire nello slot a 86 vie posto a ridosso dei drive interni del computer. I controlli sono affidati ad un dispositivo connesso alla scheda tramite un cavetto piatto, che deve giungere all'esterno del cabinet attraverso una delle piastre posteriori rimovibili. L'unità, di ingombro minimo, dispone del tradizionale **miniswitch** a levetta per attivare o escludere lo **Slow Motion**, regolabile poi tramite una **manopolina** adiacente. Per chi non avesse avuto precedenti "incontri ravvicinati" con Action Replay, si ricorda che lo Slow Motion consente di rallentare in toto l'attività di Amiga, il che può risultare utile soprattutto ai "gamofili" ad oltranza, con predilezione per i patiti di arcade: meno frenesia sul joystick (o mouse), maggiore facilità di riuscita... ed esaurimenti nervosi evitati.

In cima al "magico" scatolino, troviamo poi il rosso pulsante di **Freeze**, vera porta d'ingresso a tutte le straordinarie capaci-

tà di AR2. C'è da dire che la presenza della scheda risulta del tutto trasparente al sistema, e ci si accorge della sua presenza solo ad ogni reset, quando viene mostrata per un paio di secondi una schermata-logo graficamente più curata che nella prima versione. Ma non è certo questo il motivo che può giustificare l'acquisto di Action Replay 2.

Le prestazioni non differiscono molto da quanto già descritto sul **numero 77** della rivista, anche se in molti particolari l'evoluzione si fa sentire. Dopo il Freeze, che, come ovvio, può essere imposto mentre sta girando un qualsiasi programma o game, si entra in un ambiente esclusivo, dotato di una marea di suoi comandi il cui sunto è visualizzabile premendo il **tasto Help**. L'ambiente, tra l'altro, è ampiamente personalizzabile grazie ad una schermata di "preferences" accessibile attraverso la pressione del tasto funzione **F3**. Adoperando il mouse, si può così variare il **rapporto cromatico** sfondo/primo piano, selezionare quali **periferiche** (floppy driver) gestire, scegliere il tipo di memoria da sfruttare (solo **chip** oppure anche **fast**), nonché attivare o meno un test sulla **memoria** per intercettare eventuali interferenze di **virus**.

Questo controllo, se reso operativo, viene effettuato ogni volta che si adopera il Freeze. Anche in queste preferences, inoltre, è riservato un occhio di riguardo per chi sfrutta Amiga soprattutto per i suoi videogames: una griglia di riquadri permette infatti di abilitare l'**autofire**, con la possibilità di scegliere separatamente per 2 giocatori l'intensità di fuoco.

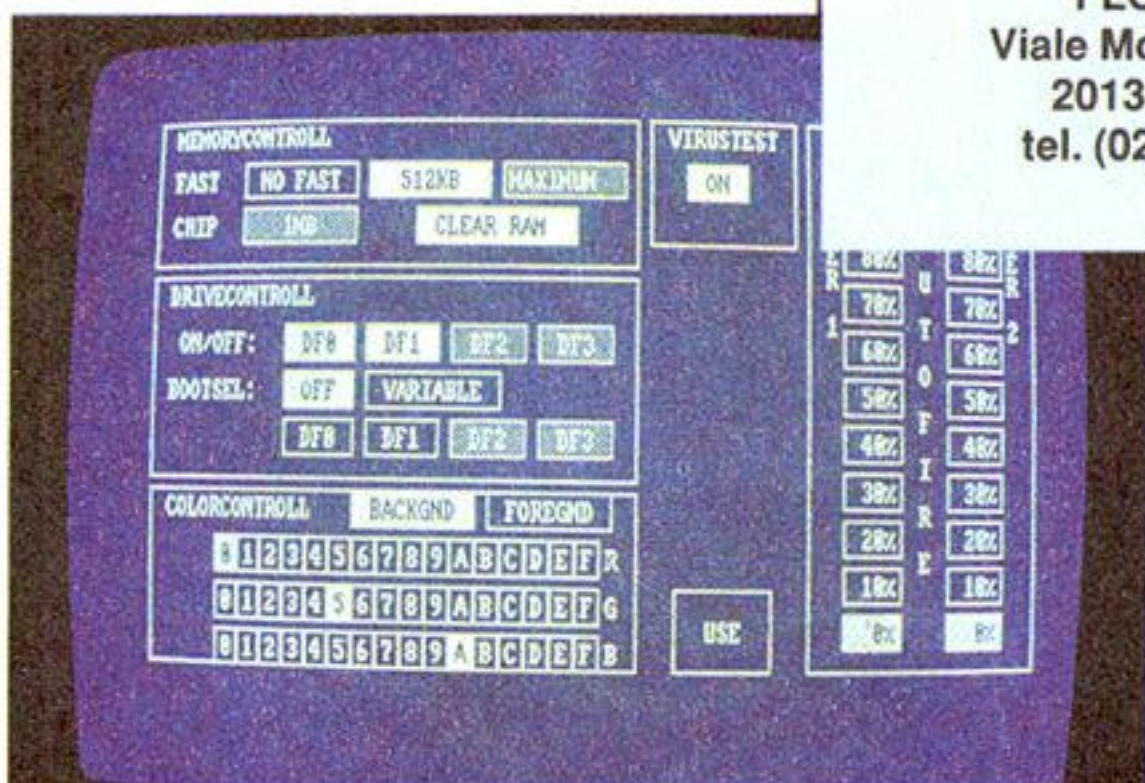
In generale, si può suddividere il raggio d'azione di Action Replay, o meglio dei suoi comandi, in due settori d'influenza. Uno rivolto a chi intende usare la scheda (o cartridge) per hackerare (eufemismo...) **programmi altrimenti incopiabili**, ed uno per smanettoni più evoluti, che sappiano muoversi agilmente nei meandri di Amiga, e che possono trovare in AR2 uno strumento insostituibile per ricavare informazioni preziose sul sistema, controllare il funzionamento di software normalmente non... scrutabile, o ancora operare modifiche "al volo" mentre un programma è in esecuzione.

Per la goduria dei primi, è possibile "grabbare" con estrema facilità **schermate Iff** eventualmente presenti in memoria, o anche brani musicali, **suoni**, eccetera. Tutto ciò che occorre fare, è impartire un comando **P** (per le schermate) oppure **Scan** e **Tracker** (per i suoni). In entrambi i casi la ricerca è automatica, con la possibilità di salvare il risultato grazie ad appositi comandi come **SP** (save picture) o semplicemente agendo su **F5** per i suoni. Inutile aggiungere che, in entrambi i casi ma soprattutto nei confronti della grafica, sono possibili numerosi rimaneggiamenti: a carico delle dimensioni dell'immagine, della loro tonalità e anche colorazione, mentre una volta adoperato Tracker per il sonoro, i tasti funzione consentono di approfondire la conoscenza con svariati standard di codifica musicale.

Per la stessa categoria di utenza, la parte del leone è comunque svolta dal Freezer vero e proprio. In pratica, una volta interrotta l'esecuzione di un programma (vogliamo dire game?) premendo il tastino rosso di Action Replay 2, si può salvare lo stesso su un normale floppy in formato Amiga digitando semplicemente **SA** seguito dal nome che intendiamo assegnargli. Tutto qua. In effetti, così facendo, lo stesso programma può poi essere ricaricato in memoria tramite il comando **LA**, il che presuppone perciò la presenza di AR2. Se si volese rendere autonoma l'esecuzione, basterà eseguire dopo il salvataggio un altro comando: **Sloader**. Questo creerà nello stesso floppy un piccolo file di nome **Aload**, che potrà essere invocato poi anche in assenza della scheda/cartuccia con la sintassi `Aload Nomefile`. Per chi non volesse sostenere l'ardua fatica di accedere al dos, è disponibile poi il comando **Install**, che renderà direttamente bootabile il programma in questione, adoperando il floppy per lo start di Amiga. Più di così....

Se la procedura appena descritta è una novità legata alla nuova versione di Action Replay, praticamente identiche sono rimaste le opzioni cosiddette di Trainer, ampiamente descritte sul numero 77. Giusto come accenno, basti ricordare che il **Trainer** consiste in una serie di comandi che permettono di fornire **"vite" infinite** ad un gioco, automatizzando quanto più possibile la ricerca in memoria delle locazioni in cui è memorizzato questo dato. Il tutto, come ovvio, accessibile anche per chi fosse ancora convinto che il linguaggio macchina corrisponda... a due clacson che dialogano tra loro.

Per chi invece ha da tempo abbandonato simili concezioni animiste (eh eh eh), Action Replay 2 mette a disposizione un set di istruzioni davvero notevole, che vanno da una completa gestione dei **drive** e del **bootblock** dei dischi, ad una

In vendita presso
FLOPPERIA
Viale Monte Nero, 15
20135 - Milano
tel. (02) 55180484

serie di informazioni supplementari disponibili sullo stato del sistema: molto più complete di quanto disponibile in ambiente Shell, e soprattutto sfruttabili in una qualunque fase di esecuzione del programma, come dire che si dispone di un **debugger** insuperabile. Per fare un esempio, si può richiedere la lista degli **interrupt attivi** (comando Interrupts), gli **indirizzi reali** delle librerie aperte, o più semplicemente l'ammontare della **memoria libera**.

Per non dire di un più tradizionale, ma super accessoriato, **Monitor di linguaggio macchina** con il quale è consentita ogni tradizionale operazione di assemblaggio/disassemblaggio, comparazioni di blocchi di memoria e loro eventuale Save, ricerca e trasferimento, e chi più ne ha più ne metta. Non manca infine un **Disk Editor** di tutto rispetto, uno **Sprite Editor** altrettanto succulento, e qualche altra feature particolare: la possibilità di codificare (e decrittare) blocchi di memoria per renderli illeggibili ad occhi indiscreti, o addirittura di **compattare/scompattare** il contenuto di sezioni di ram.

Principiante o super esperto che sia, l'acquirente di Action Replay 2 non avrà insomma di che lamentarsi: ce n'è proprio per tutti i gusti...

Videogenlock MK2+

Di Genlock abbiamo cominciato ad occuparcene nell'ultimo appuntamento. Il lettore attento saprà dunque ormai di cosa si tratta: un **apparecchio in grado di miscelare il segnale video di Amiga a quello di una fonte esterna, tipicamente un videoregistratore.**

Dopo un primo approccio rappresentato dal Minigen, eccoci ora a esaminare brevemente un prodotto dalle prestazioni analoghe, ma con qualcosa in più, e dalle origini italiane: il Videogenlock Mk2 plus, prodotto dalla **Ecr Elettronica**.

Si tratta di un apparecchio anch'esso appartenente ad una fascia intermedia di prezzo (attorno alle **400 mila**), se si considera che hardware del genere possono raggiungere costi che superano abbondantemente il milione, ma di elevata qualità, soprattutto in rapporto al fatto che gestisce anch'esso un segnale videocomposito.

L'hardware è costituito da un cabinet da affiancare al computer, fornito di un apposito cavo di collegamento Rgb. Sul retro del contenitore, infatti, sono presenti due connettori a vaschetta a 23 poli, uno femmina e uno maschio. Al primo andrà connesso il cavo video proveniente dal computer (adoperando quello a corredo), il secondo andrà invece collegato al monitor adoperando il cavo normalmente usato con il computer, per consentire la normale visualizzazione di Amiga. Un pulsante On/off posto tra i connettori, fa sì che a genlock spento il segnale di Amiga giunga come sempre al monitor, come se l'apparecchio supplementare non fosse presente. Agendo sull'interrutore, ovvero abbassando il pulsante, si attiva il genlock, e diventano così operativi gli altri due connettori (di tipo Rca) contrassegnati dalle sigle Out ed In, riservati appunto a ingresso ed uscita del segnale in videocomposito.

In pratica, anche senza adoperare il segnale di Input, si può già sfruttare il connettore di Output collegandolo ad un videotape tramite un cavo dotato di Scart, facilmente reperibile, e in simile configurazione registrare qualunque attività del computer: animazioni, schermate, eccetera.

Ma non è certo questo lo sfruttamento ideale del genlock, anche se non da ignorare. Collegando infatti un videotape all'input (adoperando lo stesso tipo di cavo con Scart da una parte e spinotti Rca dall'altra), si può accedere alle reali prestazioni di Mk2+, gestibili attraverso un selettore e una manopola presenti sul frontalino dell'apparecchio.

La levetta di selezione dispone di tre posizioni, ognuna delle quali permette tre diversi tipi di passaggio del segnale video. Ponendola su Video (al centro), al connettore Out in videocomposito giungerà il solo segnale proveniente dal videotape, mentre spostandola su Overlay (verso sinistra) si avrà la sovrapposizione totale del segnale del computer su quello video, che trasparirà attraverso lo sfondo.

Per apprezzare visivamente queste caratteristiche, è chiaro che occorrerà comunque collegare alla presa Out in videocomposito un altro videotape sul quale riversare l'effetto dei nostri esperimenti (a sua volta collegato ad un monitor), oppure effettuando la connessione con l'ingresso videocomposito di un monitor (magari lo stesso 1084 di Amiga, switchando col pulsantino frontale al modo videocomposito), in quest'ultimo caso per una mera visualizzazione del funzionamento di Mk2+.

Ma non si è ancora detto della terza posizione del selettore frontale, **Super Impose**, che in effetti è la più rara in apparecchi di questa fascia di prezzo. Questa modalità consente infatti di sovrapporre il segnale del computer a quello del video in rapporto alla posizione della manopola: ruotandola verso destra si aumenterà il livello di sovrapposizione, potendolo così tanto dosare, quanto incrementare o decrementare gradualmente fino ad **ottenere dei veri e propri effetti di dissolvenza** del segnale. Notevole.

Come ovvio, i migliori risultati si ottengono ricorrendo su Amiga a programmi di videotitolazione, mentre la qualità del risultato, mediamente molto buona, risente comunque della procedura di riversamento su un secondo videotape.

D'altro canto, come già detto, è già tanto trovarsi al cospetto di simili prestazioni ad un prezzo così contenuto, e soprattutto ad una così buona resa del segnale videocomposito, che nell'evoluzione Mk2+ del Videgenlock ha subìto una impennata qualitativa tale da poterlo considerare senza dubbio il migliore della sua fascia commerciale di appartenenza.

Cosa aggiungere, se non un... promosso a pieni voti?

di Domenico Pavone

Musica e Microdeal, il ciclo si chiude

Due soluzioni economiche, per accostarsi all'editing musicale, propopste della nota sw house

Nel corso degli ultimi appuntamenti ci si è più volte occupati di hardware e software musicale, inquadrabile in un settore che potremmo definire di medio costo e di prestazioni semiprofessionali.

Per non andare troppo lontano, solo due numeri fa si sono esaminate le notevoli caratteristiche di **Amas**, il sampler (digitalizzatore) stereo che può considerarsi uno dei prodotti di punta del settore, commercializzato dalla stessa **Microdeal** che vedremo ancora in azione in queste pagine. Pacchetti del genere, pur non raggiungendo costi astronomici, possono comunque creare qualche problema per un hobbista non troppo esigente, o che comunque non intenda sborsare somme oscillanti tra le 200 e le 400 mila lire per un uso che, tutto sommato, prevede saltuario.

Con qualche ovvia limitazione, ma senza scadere di qualità, ecco allora due prodotti di tutto rispetto volti a coprire questa fascia di utenza, accomunati da un prezzo che, senza mezzi termini, può considerarsi irrisorio, soprattutto in rapporto alle prestazioni: attorno ai 40 miseri dollari. Fate un po' voi il calcolo con il cambio corrente...

Senza perdere di vista questa realtà, diamo dunque un'occhiata ad un ennesimo pacchetto soft/hardware appartenente alla classe dei **digitalizzatori audio**, Master Sound, e a un **Sequencer / Sinthetiser** di facile uso: il programma Quartet.

Entrambi, come già detto, della Microdeal, ed entrambi imbattibili nel rapporto prezzo/qualità.

Master Sound

Molto simile come impostazione generale al più quotato (e costoso) **Amas**, Master Sound differisce dal primo soprattutto per una caratteristica: **non** è in grado di gestire contemporaneamente i due canali della stereofonia. In altre parole l'input sonoro, anche se proveniente da una fonte stereo, può essere trattato solo come **unico segnale monofonico**, globale o singolo canale che sia.

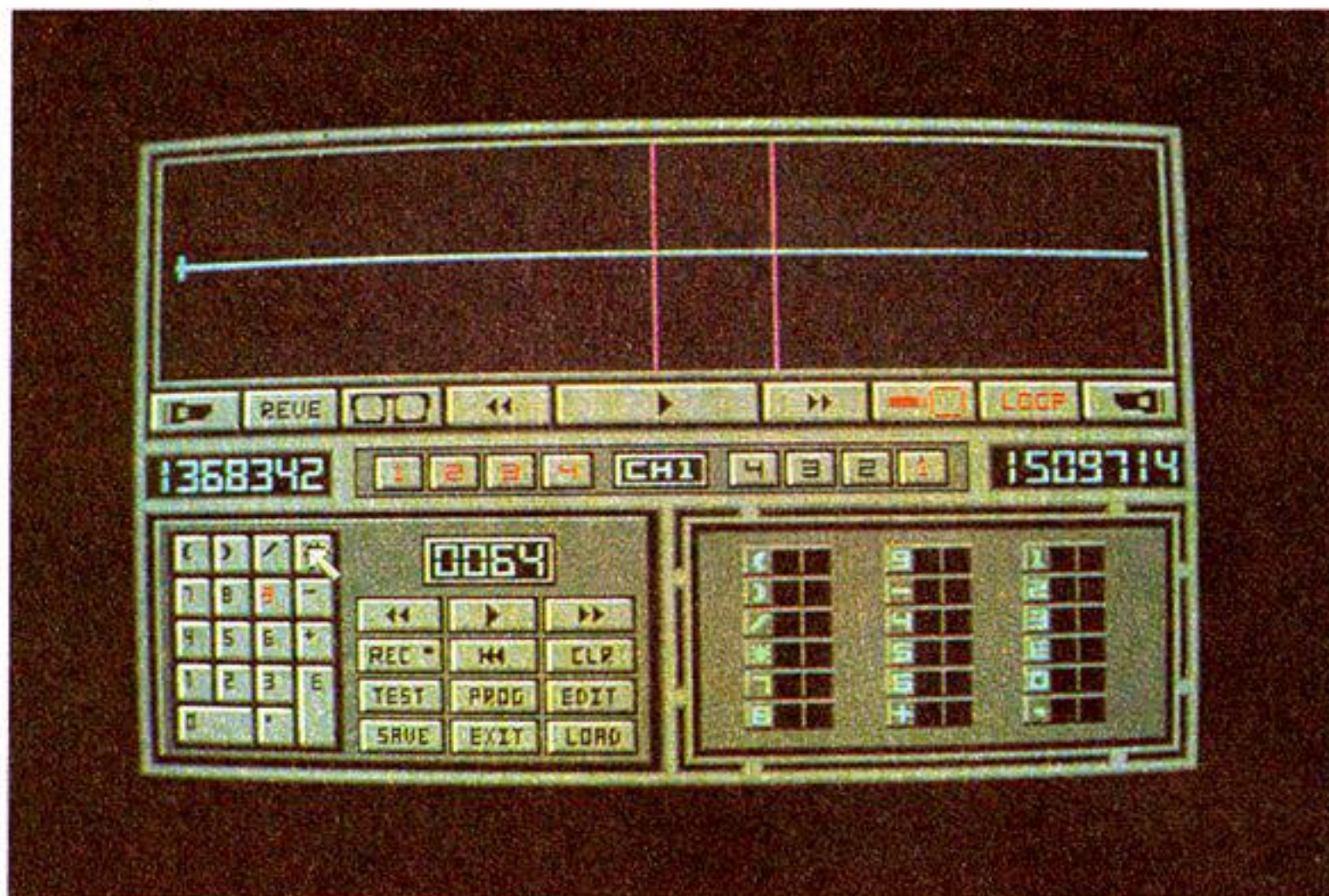
Come ovvio, il problema non si pone se si intende adoperarlo per compionamenti di suoni da "passare" poi ad un editor/sequencer, ma in effetti c'è da dire che, anche se può sembrare un grosso limite, questo è poi facilmente aggirabile. Nulla vieta, infatti, che vengano "registrati" e salvati su disco dalla fonte sonora i due singoli canali stereo, poi facilmente assemblabili in separata sede da programmi come **Audiomaster III** (vedi numero scorso della rivista), che tra l'altro risulta **totalmente compatibile** con il Master Sound.

La premessa però non tragga in inganno: la qualità del campionamento, come pure le possibilità di intervento su quanto digitalizzato, sono di livello decisamente alto, tanto da stupire per il basso costo con il quale sono realizzate.

L'hardware del package è costituito da uno scatolotto di minimo ingombro che va inserito direttamente nella **porta parallela** di Amiga, grazie ad un connettore che sporge da una delle sue estremità. All'altra è presente il connettore per l'input, che prevede l'uso di un comune mini-jack come quello a corredo di quasi tutte le cuffie da walkman. Anche se non si disponesse di un cavetto adatto, la

reperibilità di un connettore di questo tipo non rappresenterebbe certo un problema.

Il software di gestione, contenuto in **un unico floppy**, ricorda molto quello di Amas, al punto di condividerne la grafica simbolica associata alle numerose funzioni accessibili. Una volta lanciato il programma, si accede ad una schermata molto chiara e lineare, con tutte le opzioni gestibili via mouse. In alto, l'immancabile finestra riservata alla visualizzazione del suono in forma d'onda; in basso, una serie di "pulsanti" affiancati da un vasto riquadro che funziona da oscilloscopio.

Effettuati i collegamenti ed attivata la fonte sonora (Cd, Tape, Hi-fi, eccetera), Master Sound consente una regolazione ottimale del **volume** di input grazie a tre strumenti di misurazione: il già citato oscilloscopio, la simulazione software di un led a barra orizzontale con colorazione in rosso dei picchi in distorsione, ed un **analizzatore di spettro che mostra in diretta** le varie bande di frequenza. In pratica, basterà attivare una delle scelte appena accennate, e preascoltare il brano musicale (o il suono) agendo col mouse sull'apposito gadget (il primo in alto a sinistra della serie di controlli). L'ascolto avverrà come ovvio attraverso gli altoparlanti del monitor, ed è anche possibile selezionare il Playback solo su uno dei due canali o su entrambi. Lo stesso requester di scelta consente inoltre di disattivare il noto filtro taglia frequenze di Amiga, e di regolare il volume di ascolto, che

peraltro è opportuno lasciare sempre al massimo livello.

La fase di "registrazione" vera e propria, una volta effettuate queste necessarie preregolazioni, diventa poi una sciocchezza: è sufficiente attivare la fonte sonora, e clickare sul **tasto Rec**. La digitalizzazione continuerà sino a quando non si agirà nuovamente sul pulsante sinistro del mouse, o non sarà esaurita la capienza massima della finestra che visualizza la cosidetta Waveform (forma d'onda). La dimensione in byte che assumerà il corrispondente file è visualizzata subito sotto la finestra di lavoro, ed è importante prestarvi una certa attenzione. Agendo su due barre verticali presenti all'interno della finestra, è possibile infatti selezionare parte di quanto campionato, o anche decidere a priori le dimensioni del campionamento: al loro movimento (tramite il mouse) corrisponderà una modifica dei valori espressi in termini di byte, ed è importante non tendere a strafare. Disponendo di sufficiente memoria, Master Sound può digitalizzare (per esempio) anche due mega di sonoro... ma poi? A meno di disporre di hard disk, e un mare di Ram, salvare un simile file sarebbe decisamente inutile.

Nulla vieta, comunque, che si utilizzi tutta la capacità di registrazione del programma, per poi selezionare (magari dopo qualche ritocco) solo una parte del brano: le solite due barre verticali delimitano infatti anche quanto verrà salvato su periferica. Molto comodo, tra l'altro, uno **Zoom** a due posizioni (tasti a forma di occhiale e di lente da ingrandimento) che consente di espandere la visualizzazione di singole porzioni del sonoro, adoperando ancora le barre verticali per delimitare la sezione da trattare. Se le dimensioni, dopo lo zoom, eccedono i limiti della finestra visibile, sono disponibili due gadget di scorrimento laterale per visualizzare la sezione che più interessa.

Oltre al grosso **tasto di Play**, a ridosso della finestra principale sono anche disposti i "pulsanti" per il **Reverse** (ascolto invertito del brano, senza però modificarlo), ed il tasto di **Loop**, che fa riprendere

l'ascolto di quanto digitalizzato dopo la sua conclusione. Anche queste opzioni, come le altre, agiscono su quanto delimitato dalle barre verticali nella finestra principale.

Il pannello di controllo comprende poi una sezione orizzontale dedicata all'input/output (**Save, Load, Rec**) e un **tasto Iff** che consente di scegliere il formato nel quale andrà memorizzato il campionamento: il tradizionale Iff, **Raw** (dati binari inseribili in propri programmi), o come **Instrument** da adoperare eventualmente con altri programmi di elaborazione musicale.

Le ultime due serie di controlli sono poi dedicate all'editing, che consente alcune manipolazioni abbastanza interessanti come il **Fadein** e **Fadeout**, ovvero l'inserimento di un effetto di "dissolvenza" sonora ad inizio (Fadein) e/o fine (fadeout) del brano (o suono) delimitato dalle solite barre verticali. Non mancano le tradizionali funzioni di **Cut** e **Paste**, simili a quelle di un comune editor di testi, che consentono di delimitare, spostare, cancellare oppure copiare altrove sezioni della Waveform visualizzata.

L'editing, pur disponendo dele funzioni di base, non è molto sofisticato, ma, come già detto, chi volesse tentare elaborazioni più complesse può sempre ricorrere ad altro software come l'Audiomaster III.

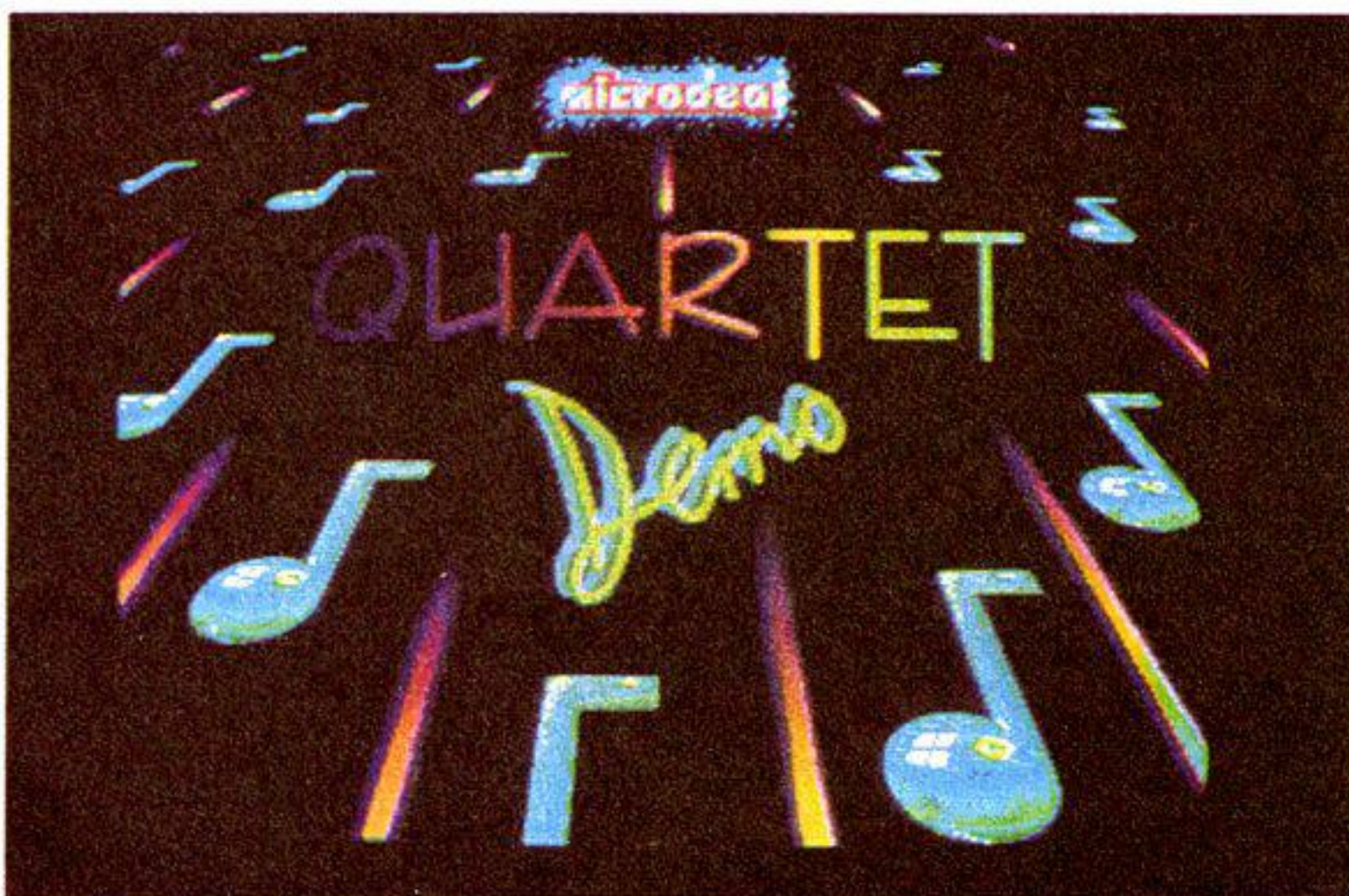
Da citare infine la capacità di caricare in memoria più file di campionamento, che possono essere poi eseguiti uno dopo l'altro grazie ad un file che memorizza solo la sequenza, non l'insieme dei brani (o suoni).

Per un costo paragonabile, se non inferiore, a quello di un videogame, in definitiva non si può che apprezzare la qualità di Master Sound, e la lungimiranza della Microdeal, che lo ha opportunamente affiancato al "cugino" Amas.

Quartet

Con questo programma ci si trova al cospetto di un più tradizionale software di composizione ed esecuzione musicale... anche se poi tanto tradizionale non è. Volendo azzardare una definizione tecnica non troppo pretenziosa, diremo che **Quartet trasforma Amiga in un sintetizzatore polifonico con recorder a 4 tracce**. Il che significa, molto semplicemente, che sarà in grado di riprodurre in contemporanea il suono di 4 strumenti campionati (da cui il nome Quartet), e naturalmente di consentire l'editing di uno spartito musicale, seppur con modalità particolari, come vedremo tra breve. L'input, per di più, può essere immesso anche tramite **Midi**, ovvero da tastiera o altro strumento dotato di tale interfaccia.

Come i musicofili più evoluti avranno già arguito, un software di queste prestazioni non può certo essere considerato un ausilio professionale, quanto piuttosto un "entry level" per chi intendesse accostarsi all'affascinante mondo della musica digitale, o anche, più banalmente, elaborare qualche brano da inserire in propri demo o startup di dischi. Valgono anche in questo caso le considerazioni fatte a proposito di Master Sound, ovvero l'assoluta competitività nel rapporto prezzo/prestazioni, cui va aggiunta la (relativa) semplicità di utilizzo, e il fatto che Quartet "giri" senza problemi anche su **Amiga 500 in configurazione base**.

Un brano musicale, come del resto avviene un po' in tutti i programmi del

genere, viene gestito da Quartet suddiviso in due componenti fondamentali: la **musica** in senso stretto, intesa come sequanza di note, e il tipo di suono (lo **strumento**) con il quale eseguirla. Ai due aspetti, da un punto di vista pratico, corrisponderanno fisicamente **due tipi di file**, uno contente la sequenza musicale, ed uno il set di strumenti ad essa associato. Un'anticipazione: adoperando due programmi inclusi nel package, questi files possono essere fatti eseguire indipendentemente da Quartet, limitatamente al sonoro oppure mentre viene visualizzata sullo schermo una immagine in formato Iff. Ovvia l'utilità di questi programmini, che consentono di inserire facilmente gli elaborati di Quartet in proprie realizzazioni, anche sofisticate. Per la cronaca, sono anche inclusi esempi di startup - sequence per ottenere "intro" musicali in propri dischetti (adatti a utenti alle prime armi), nonché sorgenti in assembler per chi sfrutta questo linguaggio.

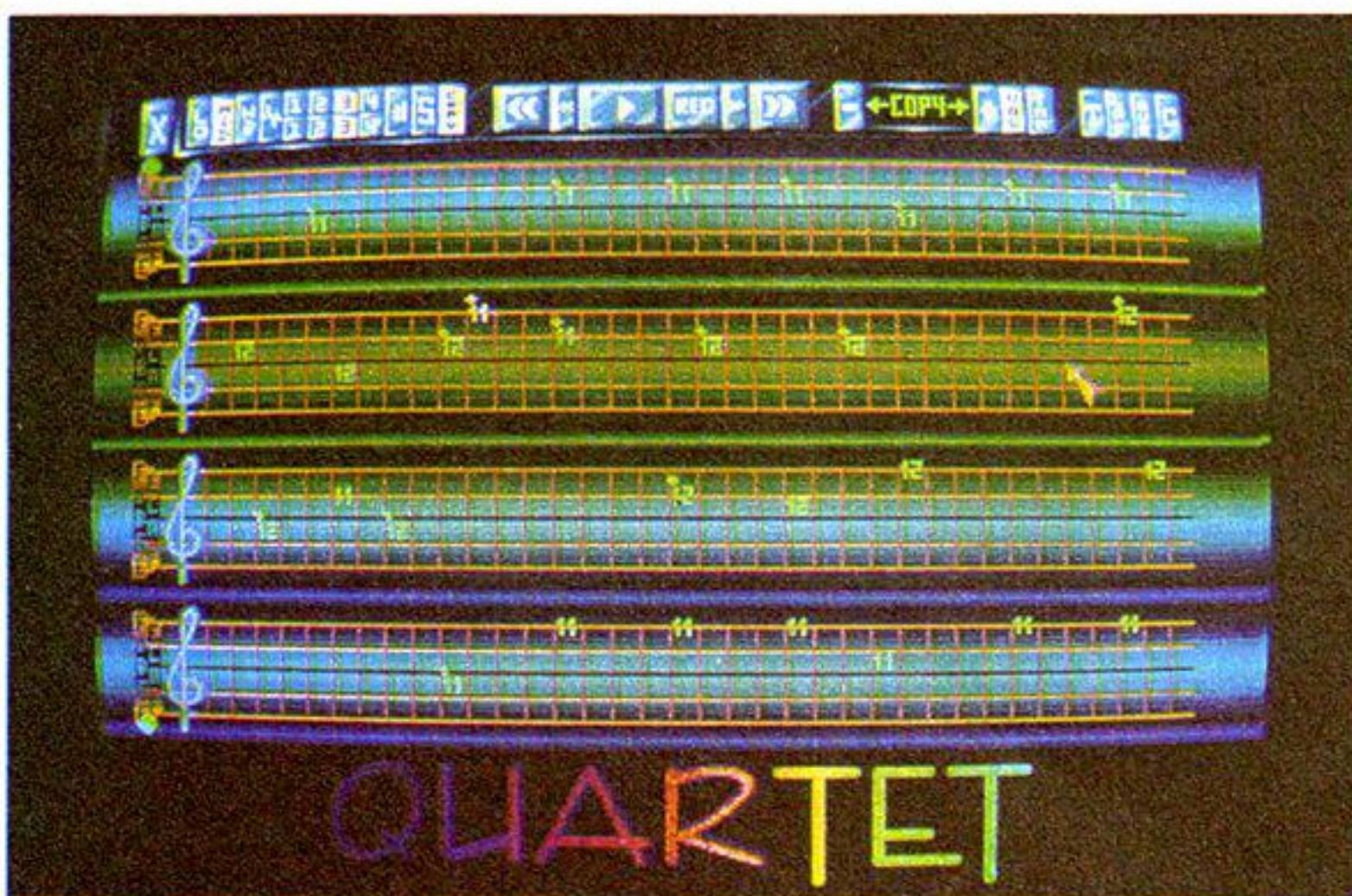
Ma procediamo con ordine, e vediamo più da vicino come opera il programma.

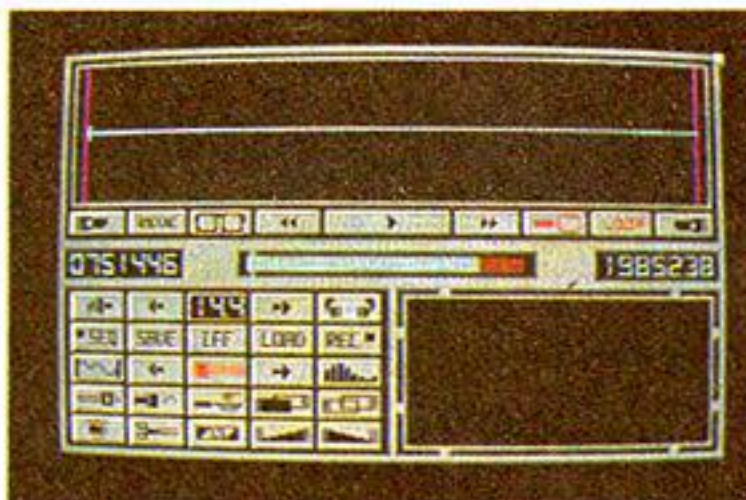
Gli ambienti di lavoro sono costituiti da due schermate. Quella cui si accede immediatamente dopo il lancio del programma, è rappresentata da un requester di input/output piuttosto particolare, circondato da una fitta serie di gadget attivabili col mouse, mentre lo sfondo si esibisce in variazioni cromatiche cangianti. Clickando su uno dei gadget-pulsanti dall'omonima intestazione, si accede al **Music Editor**, il secondo ambiente

di lavoro, caratterizzato dalla presenza di **4 pentagrammi** (le 4 tracce disponibili) e da un'altra serie di gadget posti in cima allo schermo. Una sessione di lavoro, in pratica, dovrà sempre cominciare con il caricamento da floppy degli "strumenti", costituiti da suoni campionati con un digitalizzatore, attraverso l'opzione **Load Sample** oppure **Load Set**. Con la prima, è possibile caricare singoli sample, ovvero singoli file ottenuti da un digitalizzatore, con la seconda un intero Set di **16 campionamenti**, numero massimo di sample gestibili da Quartet. I dischi a corredo del package contengono già numerosi file di entrambi i tipi, ma, come ovvio, per una massima personalizzazione è opportuno disporre di propri campionamenti. Quartet accetta per default sample ottenuti da Master Sound e Amas, ma clickando sul gadget **Sample Redo** è possibile scegliere altre frequenze di campionamento, che appaiono in evidenza all'interno del requester. Clickando su una delle scelte possibili, e poi sul **pulsante Proceed**, è assicurata la compatibilità con qualunque altro digitalizzatore audio.

In cima al requester, sono presenti due serie di pulsantini, entrambe numerate da 1 a 4, quella superiore descritta come **Block**, quella inferiore come **Sample**. Con un sistema che inizialmente può sembrare complesso, è attraverso questi pulsanti che vengono gestiti ed assegnati i 16 suoni. Ogni pulsante della riga superiore costituisce in pratica un "blocco", cui sono associati 4 differenti suoni preascoltabili agendo col mouse sui 4 gadget della riga inferiore o sfruttando il tastierino numerico laterale di Amiga; l'insieme, cosituisce il Set di strumenti. Per essere più chiari, si pensi come esempio al Block 4 come riservato alle percussioni: i 4 sample della riga inferiore potrebbero essere costituiti dal suono di 4 differenti tamburi (o piatti, o qualunque altro suono del genere si desideri).

In fase di caricamento dei sample, non sorge alcun problema se si dispone già di un set precostituito, altrimenti (per singoli campionamenti) occorrerà selezio-

nare un pulsante rappresentante un blocco, quindi uno che specifichi il sample, ed infine agire su Load Samp per caricarlo in memoria. L'insieme dei suoni, rappresentati dal nome del file corrispondente, può inoltre essere visualizzato all'interno del requester adoperando il gadget **Set** e facendo scrollare il testo con le frecce disposte lateralmente alla finestra di output. Lo stesso gadget, se selezionato con il **pulsante destro del mouse**, consente invece di determinare le dimensioni in byte dei sample, per i quali è disponibile un totale di **140 Kbyte**. Purché si faccia rientrare la dimensione totale dei 16 campionamenti in questo range, si può modificare liberamente lo spazio riservato ad ognuno di essi, in caso contrario verranno troncati per default a 10000 byte quelli dei blocchi da 1 a 3, ed a 5000 byte i suoni del quarto blocco.

Il sistema prima visto di descrizione dei suoni assume rilevanza quando si passa allo schermo di editing (al Music Editor, per intenderci).

Qui, tra gli altri, sono presenti gli stessi 8 tastini numerati in due serie da 1 a 4. Se si prova a clickare all'interno del pentagramma, apparirà un **numeretto di due cifre**, e non una nota, come ci si potrebbe aspettare. Scopo dichiarato di Quartet, infatti, è quello di consentirne l'uso anche a chi dispone solo di minime cognizioni musicali. In pratica: la prima cifra del numero rappresenta il "blocco" di cui prima si parlava, e la seconda il sample ad esso associato. Più banalmente, selezionando i pulsanti che rappresentano il sample, se ne ottiene anche l'ascolto, ed i tastini rimangono illuminati. Il numero che appare sul pentagramma corrisponderà a quei tastini illuminati, consentendo di associare uno dei 16 campionamenti a ciascuna nota, che verrà quindi eseguita con quel suono. E' questo l'aspetto forse più complesso di Quartet, ma in realtà più difficile da descrivere che da mettere in pratica.

Una volta organizzati i 16 suoni (non è comunque indispensabile adoperarli o caricarli tutti), questi possono singolarmente (a seconda di quale è selezionato al momento) subire delle modifiche elementari grazie ad un **Sound Editor** accessibile sempre dal Music Editor agendo sul gadget caratterizzato da un'onda sonora stilizzata. Dalla nuova finestra si può ascoltare il sample, metterlo in loop, nonché limitarlo ad una sua sezione o asportare delle parti, il tutto mentre viene visualizzata la forma d'onda del campionamento, e con la possibilità di adoperare uno zoom sullo stile di quello descritto nel Master Sound.

Tornando in ambiente Music Editor, ognuno dei 4 pentagrammi può essere "spento" agendo col pulsante destro del mouse mentre il puntatore si trova in testa alla traccia (all'estrema sinistra), restringendo così l'ascolto alle sole tracce desiderate. Per il Play è disponibile un tasto con la classica grafica simbolica dei registratori (singola freccia verso destra), affiancato dagli altrettanto tradizionali tasti di scorrimento (fine oppure a pagina) del testo musicale.

A questi, si aggiunge il **tasto Rec**, che consente una vera e propria registrazione di brani suonati dall'utente: adoperando la tastiera del computer, oppure, molto più proficuamente, uno strumento dotato di **output Midi** collegato ad Amiga attraverso la porta seriale. Dopo aver premuto il Rec, si può poi scegliere su quale traccia dirigere l'input clickandovi sopra, e naturalmente il tipo di sample da adoperare, agendo sui soliti 8 tastini numerati che rappresentano il Set disponibile.

Come in ogni editor musicale che si rispetti, sono inoltre disponibili alcuni automatismi facilmente implementabili con un semplice click sugli opportuni gadget: innalzamento o abbassamento di un semitono di tutto il brano (o delle sole tracce attive), modifica del tempo di esecuzione, agganciamento di note alle precedenti e scelta dell'ottava, nonché le tradizionali funzioni di Cut e Copy, che possono riguardare sezioni dello stesso pentagramma o appartenenti a differenti tracce.

Se si esclude la particolare gestione dei sample, cui peraltro ci si abitua molto in fretta, il software risulta in definitiva molto semplice ed intuitivo da usare, e quindi particolarmente adatto per un neofita non solo di Amiga, ma anche del settore musicale in genere. I risultati ottenibili, di contro, possono essere di tutta rilevanza, tanto da far guardare con assoluto rispetto a un "supereconomico" come Quartet.

di Domenico Pavone

Font Maker per Amiga, divertitevi con i caratteri

Un professionale editor di caratteri dall'italianissima Cloanto

Che cosa sia un Font Maker, chiunque abbia un attimo approfondito l'uso di Amiga non può certo ignorarlo, non fos-s'altro per la frequenza e facilità con cui questo computer ci ha abituati a trattare con caratteri di diversa impostazione grafica.

Ebbene, qualunque supposizione sul tipo di programma che ci si può aspettare non potrà che risultare limitativa, soprattutto se ci si è fatti un'idea basandosi sui numerosi editor di font presenti nel settore del pubblico dominio. Non perché questi ultimi siano da disprezzare, per carità, ma **P**ersonal **F**ont **M**aker (per gli amici **Pfm**) è davvero un'altra cosa, unico forse nel suo genere a potersi fregiare senza mezzi termini dell'aggettivo "**professionale**".

Ciò detto, va però precisato che entro certi limiti il programa può anche essere usato per la semplice creazione (o modifica) di un font da relegare tranquillamente nella sua omonima directory di sistema, ed essere utilizzato solo a video o nell'ambito di tools grafici. La potenza di Pfm consente però molto di più, soprattutto in rapporto all'**uso della stampante**. Sua caratteristica peculiare, infatti, è quella di consentire il trasferimento alle stampanti, o meglio alla loro memoria Ram (download buffer), di completi font di caratteri, con vantaggi non solo estetici o legati ad una riproduzione personalizzata, ma anche rispetto alla velocità.

Caratteri particolari, come sicuramente noto, possono infatti essere inclusi in un documento su carta solo se la stampa viene abilitata in modo grafico, con notevole dispendio di tempo e, a meno che non si disponga di hardware pregiato (leggi: laser), con risultati di non elevata qualità. O almeno, non paragonabile a quella normalmente ottenibile in modo testo (si parla di documenti, non certo di immagini).

Simili prestazioni, per quanto in parte automatizzate, richiedono comunque un discreto approfondimento dell'argomento, del resto reso possibile da un manuale di circa **300 pagine**(!) che si sofferma ampiamente sull'aspetto didattico. Non potendo certo sostituirci ad esso, scorriamo dunque velocemente, e con obbligata superficialità, le notevoli caratteristiche di Professional Font Maker.

L'editing

Dopo il lancio del programma, lo schermo si presenta con una tradizionale (o almeno in apparenza tale) griglia di editing, con alla sinistra una serie

di controlli che la Cloanto definisce "bottoni". Quando il pointer si trova all'interno della griglia, assumendo in questo caso un aspetto a croce, si può disegnare un "punto" con il pulsante sinistro del mouse, o cancellarlo con il pulsante destro. Il disegno può però essere tracciato anche in modalità brush ("pennello"), similmente a quanto implementato da tutti i tools grafici. La modalità si seleziona agendo su uno dei gadget (licenza esterofila per il solito "bottone") sulla sinistra sul quale è riprodotta una ormai standardizzata linea curva, mentre al suo fianco è disponibile un altro bottone che abilita la delimitazione di un'area della griglia, che diventerà il pennello (o brush che dir si voglia). A carico del brush, sono possibili svariate operazioni attivabili dal menu (guarda caso) di nome **Pennello**, presente nella title bar dello schermo: **lettura** e **salvataggio** da/su periferica dell'immagine del brush, sua **rotazione**, **ridimensionamento**, modifica dell'**impugnatura**, come anche modifica in **reverse** o in **corsivo** dello stesso.

Sempre adoperando i controlli laterali con grafica a bottone, l'editor consente poi di **rimpicciolire** o **ingrandire** in toto la griglia e l'immagine del carattere in essa contenuta. Modifica, questa, che interesserà l'**intero font**, non solo il singolo carattere, ma che non influirà sui reali parametri associati al font: in altre parole, una feature utile soprattutto per rendere più confortevole l'editing.

Altri automatismi resi possibili dallo stesso tipo di controlli, riguardano l'inversione dell'immagine in senso **verticale** oppure **orizzontale**, nonché lo spostamento **laterale** (o verticale) della stessa, nell'ambito della griglia di lavoro. Scontato il Clr, ovvero lo **svuotamento** della griglia, cui in caso di errore si può eventualmente ovviare con il tasto **Undo**, che ripristina la condizione preesistente all'ultima manovra effettuata.

Particolare importanza riveste la possibilità di bufferizzare l'intero carattere presente all'interno della griglia, grazie alla semplice attivazione col mouse di un bottone con freccia a sinistra, affiancato da un altro che svolge funzione opposta: riproduce nella griglia il contenuto del buffer. Importanza legata ad un aspetto cui non si è ancora fatto cenno. Finora si è parlato di un ambiente di editing, ma in realtà gli ambienti sono due, completamente autonomi, e interagibili proprio grazie alla bufferizzazione. Al di sopra dei bottoni prima descritti, è presente infatti una serie di altri gadget sotto forma di testo, sui quali si può intervenire tanto clickando con il mouse sui simboli grafici alla loro destra (per lo più delle **freccette**), quanto attivando (sempre con il mouse, ma clickando sui valori numerici) un "modo testo" che consente di immettere da tastiera la scelta desiderata.

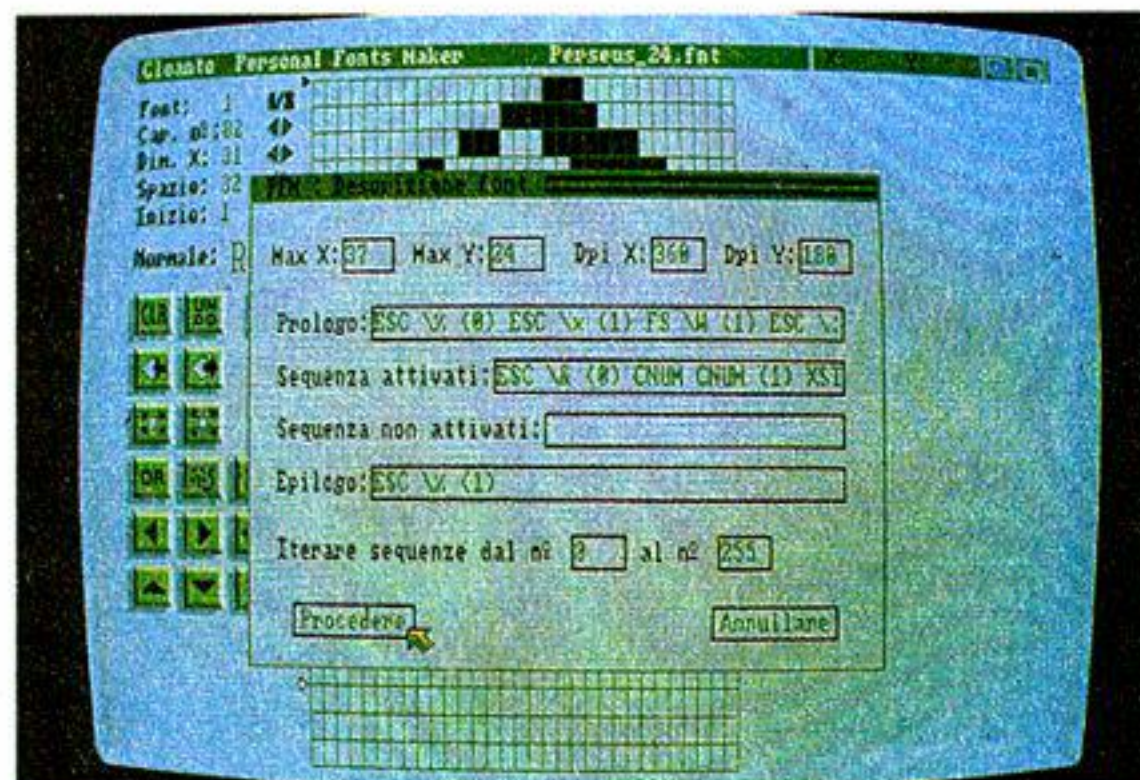
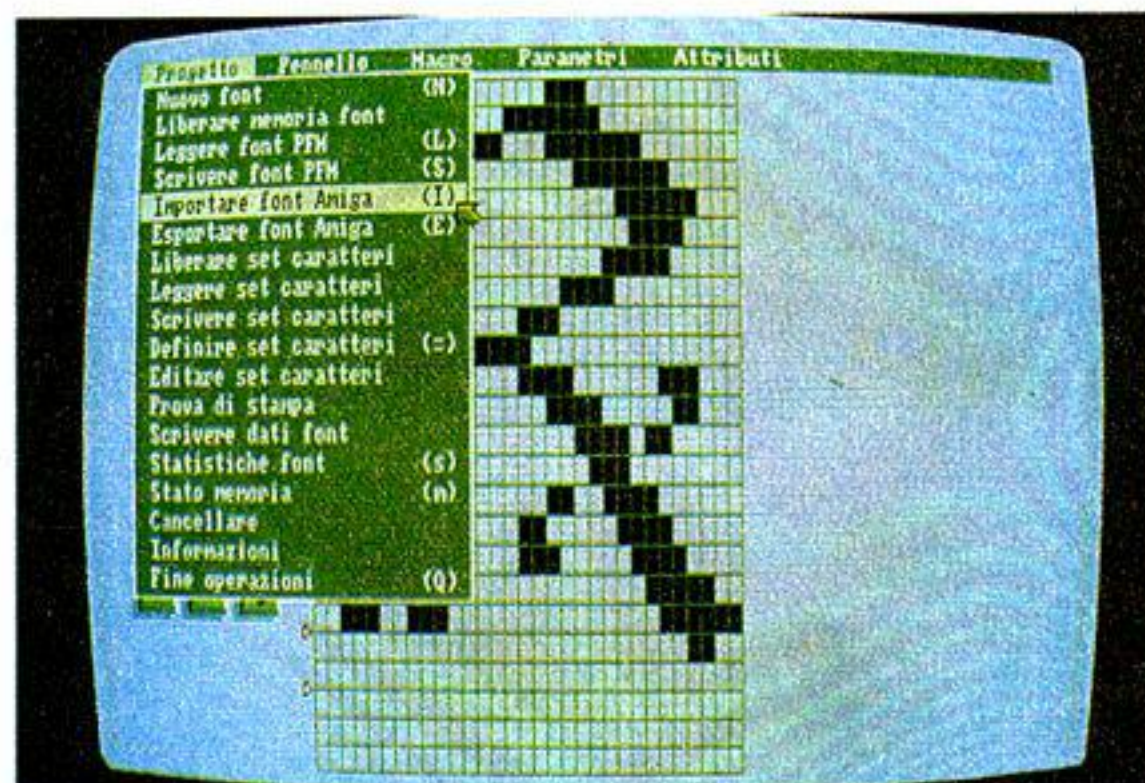
Il primo di questi elementi, anch'essi definiti come "bottoni", è specificato dalla stringa **Font**. Attivandolo, si accede ad un'altra griglia di lavoro, facente capo ad un diverso font da editare o caricare da disco. Sfruttando la bufferizzazione, si può in pratica trasferire un carattere da un ambiente all'altro, cosa molto utile se per esempio si intende apportare modifiche solo ad alcuni simboli di un certo set.

Per selezionare il carattere da editare, si può adoperare il bottone software immediatamente successivo, che opera mediante un numero di posizione del carattere nel set, oppure (più velocemente) clickando sull'opzione **Normale**, sempre presente sullo schermo. In questo caso viene visualizzato l'intero set, nel quale ci si può muovere liberamente con il mouse. Clickando sul carattere sotto il pointer, questo diventerà il campo di lavoro nella solita griglia. Quando si carica un font, non tutti i simboli visualizzati in fase di selezione rapida sono realmente attivi, come si può constatare agendo sul riquadro **Modo 1** (per accedere al modo 2), che mostrerà quelli non validi contrassegnandoli con un simbolo particolare. Questi, o meglio la loro posizione, volendo possono essere resi attivi clickandovi sopra col pulsante destro del mouse, per poi eventualmente tornare all'editing degli stessi.

Grandi manovre

Con i tre gadget successivi, si influisce sulla dimensione orizzontale dei caratteri, che influenzerà la futura visualizzazione in termini di stringa, ovvero di caratteri tra di loro adiacenti.

In particolare, agendo sul bottone **Dim.X** si può incrementare o decrementare la dimensione orizzontale della griglia contenente il carattere, che, non lo si dimentichi, viene tutta memorizzata (sotto forma di bitmap) a rappresentare detto carattere. In pratica, questa opzione andrà sfruttata se si intende elaborare un cosiddetto font **proporzionale**, in cui cioè ogni singolo elemento può avere una spaziatura diversa dagli altri. Per fare un esempio, la lettera "I" occuperà uno spazio orizzontale inferiore alla lette-

ra "**M**", tipicamente la più larga. Se il font non è proporzionale, tutti i caratteri hanno la stessa ampiezza, anche se possono essere centrati in porzioni diverse della stessa griglia.

Premesso che Professional Font Maker consente la conversione da proporzionale ad ampiezza fissa tramite una semplice macro (se ne riparlerà), non è il caso di adoperare questo "bottone" solo per aggiungere uno spazio che dividerebbe il carattere da quello successivo. A questo scopo, è infatti predisposta l'opzione **Spazio**, espressa da un valore numerico modificabile con le solite freccette. Un esempio: se la Dimesione X di un carattere risulta impostata a 30, uno Spazio di 32 implicherà la presenza di due colonne vuote a destra del carattere, anche se queste non verranno visualizzate nella griglia di lavoro. Analogamente, la funzione **Inizio** determinerà la posizione di partenza della stampa o della visualizzazione del carattere.

Se quanto finora visto riguarda soprattutto l'editing di singoli caratteri, estrema importanza riveste in Pfm la precisazione dei parametri generali riguardanti il font, le cui caratteristiche vengono associate nello stesso file che li descrive. La stessa griglia di lavoro, che poi determina tanto le dimensioni quanto la densità del carattere, va impostata per esempio tramite l'opzione del menu **Parametri/Descrizione Font**, il cui requester di input comprende le voci **Max X**, **Max Y**, **Dpi X**, **Dpi Y** (Dpi= Dot per Inch).

Nello stesso requester, appaiono voci che senza l'aiuto del manuale risulteranno incomprensibili ai più: **Prologo, Sequenza attivati, Epilogo**, eccetera. E qui, bisogna dire, le cose si fanno un po' più complicate, in quanto queste scelte determinano il formato che assumeranno i dati riguardanti i font, secondo una codifica chiamata **Ffdl**. Si tratta in pratica di un vero e proprio **linguaggio di codifica**, le cui caratteristiche possono essere approfondite (il manuale lo tratta esaurientemente), o altrimenti "bypassate" adottando dei *file Parametri* già compresi nel programma.

Va detto, inoltre, che Pfm consente di acquisire o salvare i font in un suo formato esclusivo, utile soprattutto per l'invio del font alle stampanti, oppure nel formato standard di Amiga, indispensabile, come ovvio, se si intende adoperare i caratteri per applicazioni video/grafiche gestite da altro software.

Impossibile descrivere nei dettagli la marea di facilitazioni ed opzioni legate ai menu della title bar; ci limiteremo ad accennare come ad un font possa essere direttamente assegnato un attributo selezionandolo/li dall'omonimo menu, che consta di ben 16 voci: dai comuni **Neretto**, **Corsivo**, eccetera, ai meno frequenti **Negativo**, **Bozza**, **Con Grazie**, e così via... attribuendo.

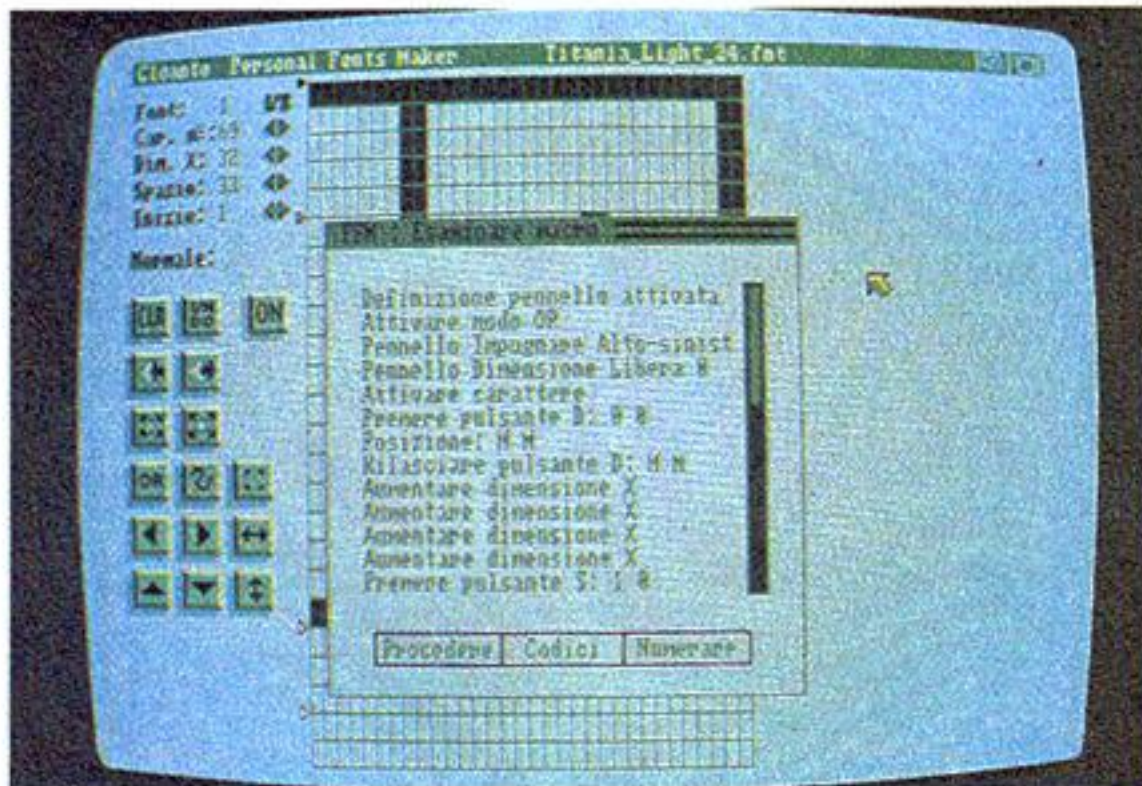
Le Macro

Prima di concludere, non si può comunque ignorare una delle caratteristiche più potenti di Personal Font Maker: l'implementazione di **Macro** in grado di svolgere, al posto dell'utente, le attività più ripetitive, o addirittura di rivoluzionare un intero font in base a precise caratteristiche comuni. Una per tutte, basti citarne una, inclusa nel package (**Shadow**) che aggiunge un'ombra a tutti i caratteri.

Alle macro è riservato un intero menu, che include alcune capacità peculiari. Intanto è possibile leggerle da disco, nel senso proprio di leggerne il contenuto, direttamente dal requester di input/output. Poiché sono memorizzate in normale testo Ascii, sono comunque esaminabili (nonché editabili) con un qualsiasi editor. Selezionata la voce **Esaminare** (del menu Macro), e successivamente il nome della macro dal requester, questa viene mostrata in forma esplicita, oppure nei suoi codici effettivi, ognuno dei quali compie una determinata azione. Questi codici sono elencati nel manuale, ma per creare una macro c'è un metodo molto più semplice ed immediato: registrarla.

Già, proprio registrarla, nel senso che si può attivare la relativa voce dal menu, assegnare un nome al file che conterrà la macro, e al solito pointer verrà aggiunta una grossa **M** per indicare che ci si trova in modalità registrazione. Da questo momento in poi, qualunque azione effettuata verrà trasformata da Pfm nel relativo codice e memorizzata nel file. La stessa andrà poi salvata su periferica (opzione **Scrivere Macro**), e, ogni volta che la si richiamerà, eseguirà esattamente le stesse manovre da noi effettuate in fase di registrazione.

Molto utili quelle già fornite a corredo del package, che consentono per esempio di dare l'**outline** ad un font, o di utilizzare un font a 7 bit come punto di partenza per convertirlo in uno ad 8 bit.

Non sarebbe finita qui, ma... è tempo di conclusioni. Che poi possono riassumersi in una sola frase: un nuovo best-seller made in Italy, da affiancare degnamente all'altro astro della Cloanto, il C1-Text.

di Roberto Corelli

Un gioco "truccato" non sempre richiede il computer

Un "trucco" per vincere (quasi) sempre al gioco reso popolare dal film "L'anno scorso a Marienbad" richiede l'impiego della matematica binaria

Il programma proposto prende spunto dall'articolo *Informatica da spiaggia* pubblicato sul numero **55** della rivista e qui proposto nelle versioni **AmigaBasic** e **Gw Basic** per Ms Dos.

In pratica si tratta del notissimo gioco della piramide di fiammiferi, costituita come in figura, giocato da due persone.

La regola fondamentale è che i due giocatori, alternativamente, possono prelevare un **qualsiasi numero** di fiammiferi, purché **da una sola riga**; vince chi prende l'ultimo fiammifero rimasto.

Esiste però un piccolo trucco col quale è possibile vincere sempre o quasi (per non togliere il piacere ai lettori di scoprirlo non viene descritto in questa sede) e tale trucco è proprio quello eseguito dal computer nel gioco riportato in queste pagine: basterà dire che esso si basa sulla numerazione binaria.

Dal momento che il computer vincerà quasi sempre, per non scoraggiare il giocatore umano esistono due livelli di gioco; nel primo il computer esegue le mosse in modo totalmente casuale, nel secondo sceglie la mossa migliore. Pertanto il primo livello è utile per ambientarsi nell'ottica del gioco.

```
      1
     1 1
    1 1 1
   1 1 1 1
  1 1 1 1 1
 1 1 1 1 1 1
1 1 1 1 1 1 1
```

La piramide di fiammiferi

Come si gioca

Dopo aver scelto il livello, appare una semplice schermata contenente, al centro, l'ormai celebre piramide; alla sua sinistra i numeri da **1** a **7** riportano, in colonna, il numero di riga; le lettere da **A** a **M** (che compaiono in alto) indicano, invece, la colonna. A destra della piramide viene visualizzata una misteriosa(!) sfilza di numeri binari.

Compare, quindi, la richiesta...

Quale riga (1-7)?

...che chiede da quale riga si vorranno prelevare i fiammiferi dalla piramide.

La domanda successiva...

Quale unità (/=fine)?

...chiede la colonna da cui si intende prelevare (nella riga scelta precedentemente) tramite la digitazione (in minuscolo!) di una lettera da A a M. Digitando la barra inclinata (/) si passa il turno al cervellone.

E' come se si giocasse a battaglia navale, in cui si danno le coordinate della nave da affondare.

Da notare che, dopo ogni mossa (nella colonna di sinistra) appare il numero dei fiammiferi rimasti nella piramide, oltre a vari messaggi nel caso si indicasse, per errore, una riga o una casella vuota.

Particolarmente utile è la subroutine denominata **Binary** (per l'Amiga) e quella che inizia alla linea **1360** (per Ms Dos) che convertono un numero qualsiasi da decimale a binario: variando il valore della variabile **PS** si varia il range dei numeri convertibili, secondo lo schema seguente:

PS	N. dec. Convertibili
3	0 < n < 7
4	0 < n < 15
5	0 < n < 31
6	0 < n < 63

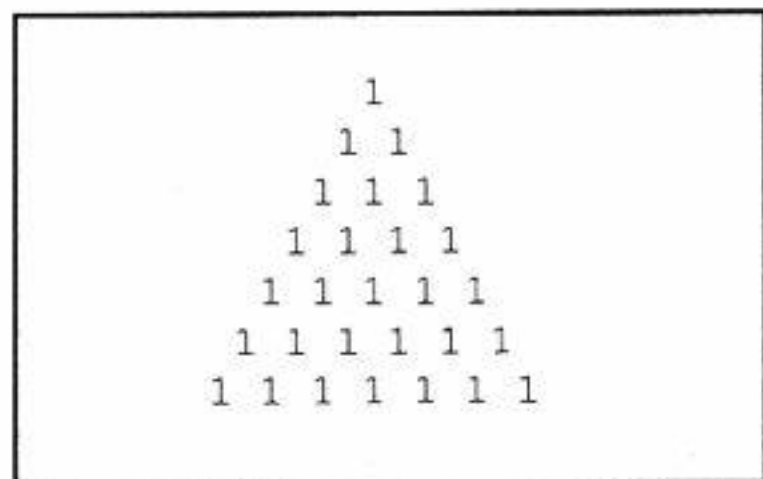
...eccetera.

Ma che diavolo saranno mai quei numeri binari sulla destra che cambiano continuamente dopo ogni mossa?

Non sono altro che la "traduzione", in base binaria, del numero intero decimale (compreso tra 0 e 7).

Tutti(!) dovrebbero sapere che tale traduzione corrisponde, limitatamente ai primi otto valori, a...

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Ora, senza entrare nel merito della matematica binaria, il trucco per vincere al gioco consiste nel fare in modo che le tre file verticali di zero ed uno contengano un numero pari di simboli 1.

Appena il gioco inizia, quindi, le tre file contengono già un numero pari di simboli 1 (ogni colonna ne contiene, infatti, 4).

Seguendo le regole del gioco, pertanto, bisogna eliminare (da un'unica fila, lo ricordiamo) un numero tale di fiammiferi tale che il numero simboli 1 (conteggiato verticalmente) rimanga sempre pari. Per esempio, se ad un certo punto del gioco la situazione è come quella seguente...

```
0   000
1   001
2   010
3   011
4   100
4   100
2   010
7   111
```

...(in cui, cioè, vi sono **tre** 1 nella prima fila verticale a sinistra, **quattro** 1 in quella centrale e **tre** 1 a destra) eliminando dalla quinta fila **tre** simboli 1 (lasciando, quindi, un solo fiammifero) riporteremo la "parità" nelle tre file verticali.

Se, procedendo nel modo descritto, si costringe l'avversario ad arrivare fino alla fine presentandogli sempre colonne "pari", si ottiene la vincita.

A mano a mano che vengono eliminati i fiammiferi (è infatti possibile toglierne solo uno alla volta) la "traduzione" in codice binario viene aggiornata immediatamente, in modo da consentire alcuni ragionamenti. Ovvio che tale opportunità è stata inserita per puro scopo didattico.

Da notare che le combinazioni possibili per raggiungere la parità sono numerose; alcune, però, riescono ad inchiodare l'avversario; altre, invece, gli lasciano qualche scappatoia aperta.

Inutile dire che, chi lo desidera, può apportare numerose modifiche, tra cui suggeriamo la possibilità di sistemare nuovamente alcuni fiammiferi (eliminati per errore) prima di passare la mano al computer.

Grazie alla semplicità della procedura, il gioco può essere giocato anche senza l'aiuto del computer; è però indispensabile imparare a memoria le "traduzioni" ed effettuare a mente i vari calcoli: un buon esercizio di memoria, non c'è che dire!

```
' ---     The binary game      ---
' ---    by Roberto Corelli   ---
' ---     versione AmigaBasic ---
DEFINT a-z: RANDOMIZE TIMER
' BIN$ ()     --> numeri binari da 1 a 7
' AMITAB () --> scacchiera del computer
' BOX ()      --> per il secondo livello di gioco
' MOSSE$ () --> mosse valide al secondo livello
DIM bin$ (7), amitab (7, 13), box (7, 13), mosse$
(7)
riga$ = STRING$ (30, " "): pezzi = 28
SCREEN 1, 320, 200, 1, 1: WINDOW 1, "The binary
game", , 0, 1
WHILE a$ <> "1" AND a$ <> "2"
  a$ = INKEY$
  LOCATE 10, 5: PRINT "Scegli livello  (1-2)"
WEND
lv = VAL (a$): CLS: GOSUB GameField
' --- loop principale ---
WHILE pezzi <> 0
  player = 0: amiga = 0
  GOSUB PlayerMove
  IF player = 1 AND pezzi = 0 THEN GOTO PlayerWins
  GOSUB AmigaPlays
WEND
PlayerWins:
IF player THEN CLS: BEEP: PRINT "Hai vinto!"
IF amiga THEN CLS: BEEP: PRINT "Ho vinto!"
answer$ = "": PRINT "Un'altra partita?  (s/n)"
WHILE answer$ <> "s" AND answer$ <> "n"
  answer$ = INKEY$
WEND
IF answer$ = "s" THEN RUN
CLS: PRINT "Bye! Bye!": END
' ---- subroutines ---
PlayerMove:
LOCATE 17, 7: INPUT "Quale riga  (1-7)"; x: c = 0
' *** conta quante caselle '1' ci sono in riga X
***
FOR y = 1 TO 13
  IF amitab (x, y) = 1 THEN c = c + 1
NEXT y
' *** la riga X e' vuota ***
IF c = 0 THEN
  z$ = "la riga " + STR$ (x) + " e' vuota!"
  CALL errore  (z$, riga$): GOTO PlayerMove
END IF
Label1:
LOCATE 19, 3: INPUT "Quale unita'   (/ = fine)";
unit$
IF unit$ = "/" THEN GOTO Fine
' *** immettere solo lettere minuscole,    ***
' *** dal momento che la lettera immessa ***
' *** viene convertita in maiuscolo  e     ***
' *** trasformata nell'equivalente codice***
' *** ASCII per trarre il numero   di      ***
' *** colonna della scacchiera  (A-M)      ***
y$ = UCASE$ (unit$): y1 = ASC (y$)-64
' *** mette uno '0' riga X colonna Y1 se ***
' *** la colonna contiene '1'             ***
IF amitab (x, y1) = 1 THEN
  LOCATE x + 6, y1 + 10: PRINT "0"
ELSE
   ' *** la casella indicata e' vuota ***
  LOCATE 1, 1: PRINT "La casella indicata e' vuota!"
  FOR z = 1 TO 8000: NEXT z
  LOCATE 1, 1: PRINT riga$: GOTO Label1
END IF
' *** trasforma in binario il numero delle ***
' *** caselle contenenti '1' in riga X     ***
amitab (x, y1) = 0: CALL binary ( (x), amitab ())
pezzi = pezzi-1: bin$ (x) = bin$
LOCATE 2, 1: PRINT "Pezzi rimasti -"; pezzi: GOTO
Label1
Fine:
' *** svuota le linee contenenti le richieste ***
FOR i = 1 TO 2: LOCATE 16 + i, 1: PRINT riga$: NEXT
i
player = 1: RETURN
' ----------
```

```
AmigaPlays:
LOCATE 1, 1: PRINT "Ok. Ora tocca a me!"
ON lv GOSUB Level1, Level2
LOCATE 1, 1: PRINT riga$: RETURN
' ----------
Level1:
' X --> num. riga
' Y --> num. colonna
' PZ -> num. di pezzi da prelevare in riga X
GOSUB verifica
FOR volte = 1 TO pz
  Loop2:
  ' *** prende una colonna a caso da riga X ***

  y =  (RND (1)*13) + 1
  ' *** se la casella X, Y contiene '1' azzerala
***
  IF amitab (x, y) = 1 THEN
    LOCATE x + 6, y + 10: PRINT "0"
  ELSE
    GOTO Loop2
  END IF
  ' *** trasforma il numero delle caselle in riga
X ***
  ' *** contenenti '1' nel corrispondente binario
***
  amitab (x, y) = 0: CALL binary ( (x), amitab ())
  pezzi = pezzi-1: bin$ (x) = bin$
  LOCATE 2, 1: PRINT "Pezzi rimasti -"; pezzi
NEXT volte: amiga = 1
LOCATE 17, 1: PRINT riga$: RETURN
' ----------
Level2:
sx = 1: GOSUB verifica
' *** copia la riga X in BOX () ***
FOR z = 1 TO 13: box (x, z) = amitab (x, z): NEXT
z
FOR volte = 1 TO pz
  Loop3:
  ' *** preleva una colonna a caso da riga X ***
  y = RND (1)*13: IF y = 0 THEN y = 1
  ' *** se la casella X, Y e' piena la mette in
MOSSE$ () ***
 ' *** (se la mossa e' valida, vengono effettuate
le   ***
    '  ***  mosse  contenute  in  tale  matrice
***
  IF amitab (x, y) = 1 THEN
     amitab (x, y) = 0
     a$ = STR$ (x): b$ = STR$ (y): z = 1: IF LEN
(b$) = 3 THEN z = 2
     mosse$ (volte) = RIGHT$ (a$, 1) + MID$ (b$,
2, z)
  ELSE
     GOTO Loop3
  END IF
NEXT volte
' *** trasforma il numero delle caselle contenenti
'1' ***
'  ***  in  riga  X  nel  corrispondente  binario
***
CALL binary ( (x), amitab ()): bin$ (x) = bin$
c1 = 0: c2 = 0: c3 = 0
' *** conta quanti '1' binari ci sono nelle tre
colonne ***
FOR z = 1 TO 7
  b$ = bin$ (z)
  IF LEFT$ (b$, 1) = "1" THEN c1 = c1 + 1
  IF MID$ (b$, 2, 1) = "1" THEN c2 = c2 + 1
  IF RIGHT$ (b$, 1) = "1" THEN c3 = c3 + 1
NEXT z
' *** se SW = 1 significa che il numero degli '1'
binari ***
'    ***    nelle    tre    colonne    e'    pari
***
GOSUB control: IF sw = 1 THEN GOTO ok
' *** mossa errata;  riprendi il contenuto della
***
' *** vecchia riga X                          ***
FOR z = 1 TO 13: amitab (x, z) = box (x, z): NEXT
z
' *** ricalcola il vecchio valore binario ***
CALL binary ( (x), amitab ())
bin$ (x) = bin$: GOTO Level2
ok:
LOCATE 17, 2: PRINT "Io muovo"; pz; " pezzi da
riga"; x
FOR z = 1 TO 8000: NEXT z
' *** trasforma le stringhe di MOSSE$ () ***
' *** in numeri per l'effettuazione     ***
' *** pratica delle mosse               ***
FOR p = 1 TO pz
  x = VAL (LEFT$ (mosse$ (p), 1))
  y = VAL (RIGHT$ (mosse$ (p), 1))
 IF LEN (mosse$ (p)) = 3 THEN y = VAL (MID$ (mosse$
(p), 2, 2))
  LOCATE x + 6, y + 10: PRINT "0"
 pezzi = pezzi-1: LOCATE 2, 1: PRINT "Pezzi rimasti
-"; pezzi
NEXT p: amiga = 1
LOCATE 17, 1: PRINT riga$: RETURN
' ----------
GameField:
' *** mette 28 '1' nella scacchiera del computer
***
p = 7: q = 1: s = 13
WHILE p> = 1
  FOR y = q TO s STEP 2
    amitab (p, y) = 1
    LOCATE p + 6, y + 9: PRINT amitab (p, y)
  NEXT y
  p = p-1: q = q + 1: s = s-1
WEND
LOCATE 5, 11: PRINT "ABCDEFGHIJKLM"
' *** a sinistra della scacchiera scrive il numero
di riga
' *** e a destra i relativi numeri binari ***
FOR y = 1 TO 7
  LOCATE y + 6, 8: PRINT CHR$ (48 + y)
  LOCATE y + 6, 26: PRINT bin$ (y)
NEXT y
FOR x = 1 TO 7
  CALL binary ( (x), amitab ()): bin$ (x) = bin$
NEXT x
RETURN
' ----------
```

```
control:
' *** se tutte e tre le colonne contengono un ***
' *** numero pari di '1' binari,  allora SW = 1
***
sw = 0
IF c1 = 0 OR c1 = 2 OR c1 = 4 OR c1 = 6 THEN
  IF c2 = 0 OR c2 = 2 OR c2 = 4 OR c2 = 6 THEN
    IF c3 = 0 OR c3 = 2 OR c3 = 4 OR c3 = 6 THEN
      sw = 1
    END IF
  END IF
END IF
RETURN
verifica:
c = 0
' *** preleva una riga a caso e       ***
' *** controlla che non sia vuota     ***
' *** in tal caso preleva altra riga ***
WHILE c = 0
  x = RND (1)*7
  FOR v = 1 TO 13
    IF amitab (x, v) = 1 THEN c = c + 1
  NEXT v
WEND
pz = RND (1)*c: IF pz = 0 THEN pz = 1
IF sx = 1 THEN sx = 0: RETURN
LOCATE 17, 2: PRINT "Io muovo"; pz; " pezzi da
riga"; x
FOR z = 1 TO 8000: NEXT z
RETURN

' ---- sottoprogrammi ---
SUB errore (a$, b$) STATIC
   BEEP: LOCATE 1, 1: PRINT a$
   FOR z = 1 TO 8000: NEXT z
   LOCATE 1, 1: PRINT b$
END SUB

SUB binary  (a, a (2)) STATIC
  SHARED bin$
  c = 0: ps = 3: bin$ = "000"
  ' *** conta quante caselle '1' ci sono in riga X
***
  FOR v = 1 TO 13
    IF a (a, v) = 1 THEN c = c + 1
  NEXT v
  WHILE c>0
    IF c = 2 OR c = 4 OR c = 6 THEN MID$ (bin$, ps,
1) = "0"
    IF c = 1 OR c = 3 OR c = 5 OR c = 7 THEN MID$
(bin$, ps, 1) = "1"
    c = INT (c/2): ps = ps-1
  WEND
  LOCATE a + 6, 26: PRINT bin$
END SUB
```

```
10 REM ---    The binary game    ---
20 REM ---   by Roberto Corelli  ---
30 REM --- versione  per  MSDOS  ---
35 DEFINT a-z: RANDOMIZE TIMER
40 DIM bin$ (7), comptab (7, 13), box (7, 13),
mosse$ (7)
50 riga$ = STRING$ (30, " "): pezzi = 28
60 WHILE a$ <> "1" AND a$ <> "2"
70 a$ = INKEY$
80 LOCATE 10, 25: PRINT "Scegli livello  (1-2) "
90 WEND
100 lv = VAL (a$): CLS: GOSUB 1070
110 REM --- loop principale ---
120 WHILE pezzi <> 0
130 player = 0:  ibm = 0
140 GOSUB 260: IF player = 1 AND pezzi = 0 THEN
GOTO 170
150 GOSUB 440
160 WEND
170 IF player THEN CLS: BEEP: PRINT "Hai vinto!"
180 IF ibm THEN CLS: BEEP: PRINT "Ho vinto!"
190 answer$ = "": PRINT "Un'altra partita?  (s/n)
"
200 WHILE answer$ <> "s" AND answer$ <> "n"
210 answer$ = INKEY$
220 WEND
230 IF answer$ = "s" THEN RUN
240 CLS: PRINT "Bye! Bye!": END
250 REM ---- subroutines ---
260 REM Muove il giocatore
270 LOCATE 17, 27: INPUT "Quale riga  (1-7) "; x:
c = 0
280 FOR y = 1 TO 13: IF comptab (x, y)  = 1 THEN c
= c + 1
300 NEXT y
310 IF c = 0 THEN z$ = "la riga " + STR$ (x)  + "
e' vuota!":  GOSUB 1310:  GOTO 270
320 LOCATE 19, 23: INPUT "Quale unita'  (/ = fine)
"; unit$
330 IF unit$ = "/" THEN GOTO 420
335 IF unit$ <"a" OR unit$> "m" THEN GOTO 320
340 y$ = UCASE$ (unit$): y1 = ASC (y$) -64
350 IF comptab (x, y1)  = 1 THEN LOCATE x + 6, y1
+ 30: PRINT "O":  GOTO 390
360 LOCATE 1, 25: PRINT "La casella indicata e'
vuota!"
```

```
370 FOR z = 1 TO 8000: NEXT z
380 LOCATE 1, 25: PRINT riga$: GOTO 320
390 comptab (x, y1)  = 0:  GOSUB 1360
400 pezzi = pezzi-1: bin$ (x)  = bin$
410 LOCATE 2, 31: PRINT "Pezzi rimasti -"; pezzi:
GOTO 320
420 FOR i = 1 TO 3: LOCATE 16 + i, 21: PRINT riga$:
NEXT i
430 player = 1: RETURN
440 REM ----------
450 REM Gioca il computer
460 LOCATE 1, 31: PRINT "Ok. Ora tocca a me!"
470 ON lv GOSUB 490,  690
480 LOCATE 1, 25: PRINT riga$: RETURN
490 REM ----------
500 REM Level1
600  GOSUB  1460:  FOR  volte  =  1  TO  pz

610 y = RND (1)  * 13
620 IF comptab (x, y)  = 1 THEN LOCATE x + 6, y +
30: PRINT "0":  GOTO 640
630 GOTO 610
640 comptab (x, y)  = 0:  GOSUB 1360
650 pezzi = pezzi-1: bin$ (x)  = bin$
660 LOCATE 2, 31: PRINT "Pezzi rimasti -"; pezzi
670 NEXT volte:  ibm = 1
680 LOCATE 17, 25: PRINT riga$: RETURN
690 REM ----------
700 REM Level2
750 sx = 1: GOSUB 1460
780 FOR z = 1 TO 13: box (x, z)  = comptab (x, z):
NEXT z
790 FOR volte = 1 TO pz
800 y = INT (RND (1)  * 13): IF y = 0 THEN y = 1
810 IF comptab (x, y)  = 0 THEN GOTO 800
820 comptab (x, y)  = 0
825 a$ = STR$ (x): b$ = STR$ (y): z = 1: IF LEN
(b$)  = 3 THEN z = 2
830 mosse$ (volte)  = RIGHT$ (a$, 1)  + MID$ (b$,
2, z)
850 NEXT volte
860 GOSUB 1360: bin$ (x)  = bin$: c1 = 0: c2 = 0:
c3 = 0
880 FOR z = 1 TO 7: b$ = bin$ (z)
890 IF LEFT$ (b$, 1)  = "1" THEN c1 = c1 + 1
900 IF MID$ (b$, 2, 1)  = "1" THEN c2 = c2 + 1
910 IF RIGHT$ (b$, 1)  = "1" THEN c3 = c3 + 1
920 NEXT z: GOSUB 1230: IF sw = 1 THEN GOTO 970
940 FOR z = 1 TO 13: comptab (x, z)  = box (x, z):
NEXT z
950 GOSUB 1360: bin$ (x)  = bin$: GOTO 700
970 LOCATE 17, 25: PRINT "Io muovo"; pz; " pezzi
da riga"; x
980 FOR z = 1 TO 8000: NEXT z
990 FOR p = 1 TO pz
1000 x = VAL (LEFT$ (mosse$ (p), 1) )
1010 y = VAL (RIGHT$ (mosse$ (p), 1) )
1020 IF LEN (mosse$ (p) )  = 3 THEN y = VAL (MID$
(mosse$ (p), 2, 2) )
1030 LOCATE x + 6, y + 30: PRINT "0"
1040 pezzi = pezzi-1: LOCATE 2, 31: PRINT "Pezzi
rimasti -"; pezzi
1050 NEXT p:  ibm = 1: LOCATE 17, 31: PRINT riga$:
RETURN
1070 REM ----------
1080 REM GameField
1090 p = 7: q = 1: s = 13
1100 WHILE p>  = 1
1110 FOR y = q TO s STEP 2: comptab (p, y)  = 1
1130 LOCATE p + 6, y + 29: PRINT comptab (p, y):
NEXT y
1140 p = p-1: q = q + 1: s = s-1
1150 WEND
1160 LOCATE 5, 31: PRINT "ABCDEFGHIJKLM"
1170 FOR x = 1 TO 7: GOSUB 1360: bin$ (x)  = bin$:
NEXT x
1180 FOR x = 1 TO 7
1190 LOCATE x + 6, 28: PRINT CHR$ (48 + x)
1200 LOCATE x + 6, 46: PRINT bin$ (x)
1210 NEXT x
1220 RETURN
1230 REM ----------
1240 REM controllo mossa
1250 sw = 0
1260 IF c1 <> 0 AND c1 <> 2 AND c1 <> 4 AND c1 <>
6 THEN GOTO 1300
1270 IF c2 <> 0 AND c2 <> 2 AND c2 <> 4 AND c2 <>
6 THEN GOTO 1300
1280 IF c3 <> 0 AND c3 <> 2 AND c3 <> 4 AND c3 <>
6 THEN GOTO 1300
1290 sw = 1:  RETURN
1300 RETURN
1310 REM ----------
1320 REM errore!
1330 BEEP: LOCATE 1, 25: PRINT z$: FOR z = 1 TO
8000: NEXT z
1340 LOCATE 1, 25: PRINT riga$:  RETURN
1350 REM ----------
1360 REM conversione decimale-binario
1370 c = 0: ps = 3: bin$ = "000"
1380 FOR v = 1 TO 13: IF comptab (x, v)  = 1 THEN
c = c + 1
1390 NEXT v
1400 WHILE c> 0
1410 IF c = 2 OR c = 4 OR c = 6 THEN MID$ (bin$,
ps, 1)  = "0"
1420 IF c = 1 OR c = 3 OR c = 5 OR c = 7 THEN MID$
(bin$, ps, 1)  = "1"
1430 c = INT (c/2): ps = ps-1
1440 WEND
1450 IF flag = 1 THEN RETURN
1455 LOCATE x + 6, 46: PRINT bin$:  RETURN
1460 REM ----------
1470 c = 0
1480 WHILE c = 0
1490 x = RND (1)  * 7: FOR v = 1 TO 13
1500 IF comptab (x, v)  = 1 THEN c = c + 1
1510 NEXT v
1520 WEND
1530 pz = RND (1)  * c: IF pz = 0 THEN pz = 1
1540 IF sx = 1 THEN sx = 0: RETURN
1550 LOCATE 17, 25: PRINT "Io muovo"; pz; " pezzi
da riga"; x
1560 FOR z = 1 TO 8000: NEXT z: RETURN
```

di Dario Pistella

Giochiamo a briscola con il C/64, Ms-Dos, Amiga

Uno dei giochi più semplici da implementare è anche uno dei più noti in Italia

Il programma di queste pagine (valido per qualsiasi computer Commodore), non è altro che l'implementazione di uno dei più famosi e diffusi giochi di carte: la briscola.

Briscola, come si gioca

Nel gioco della briscola si utilizza un mazzo di 40 carte; nella prima mano vengono distribuite tre carte per ogni giocatore, quindi ne viene pescata un'altra che verrà posta sotto il mazzo e che sarà la "Briscola", una specie di Jolly.

A questo punto uno dei giocatori mette sul tavolo una carta, che l'avversario potrà:

Prendere, con una carta che abbia lo stesso seme e valore superiore, o con una carta che abbia lo stesso seme della briscola; se la carta in tavola è una briscola, potrà essere presa solo con un'altra briscola di valore maggiore.

Lasciare, con una carta che abbia lo stesso seme e valore inferiore o con un qualsiasi seme diverso da quello della carta in tavola e un qualsiasi valore. Tutte le carte valgono **0** punti, ad eccezione di: 8, che vale **2** punti; 9, che vale **3** punti; 10, che vale **4** punti; 3, che vale **10** punti; 1 (asso), che vale **11** punti.

Ogni volta che si conclude un'operazione (ogni volta che, cioè, entrambi i giocatori hanno giocato una carta), il giocatore che ha preso le carte dal tavolo pesca per primo dal mazzo, subito seguito dall'altro giocatore.

La briscola viene considerata come una normale carta, ma viene posta in fondo al mazzo. A mazzo terminato, si continua fino ad esaurimento delle carte. Scopo del gioco è quello di ottenere più punti del proprio avversario. Sarà utile ricordare che il totale dei punti disponibili è **120**, quindi per vincere è sufficiente totalizzare almeno 61 punti.

Giocando

Durante il gioco il computer mostra in alto a sinistra le carte a disposizione dell'utente; in basso, invece, viene indicata la briscola, le carte rimaste nel mazzo ed i punti totalizzati fino a quel momento dal computer e dal giocatore, in modo da avere, in ogni momento, la situazione sotto controllo.

A mazzo esaurito, le carte usate, e non rimpiazzate, verranno sostituite con uno 0. Per comunicare al computer la propria scelta, è sufficiente digitare il numero della carta (corrispondente a quella visualizzata sullo schermo) che si intende giocare (cioè 1, 2, 3).

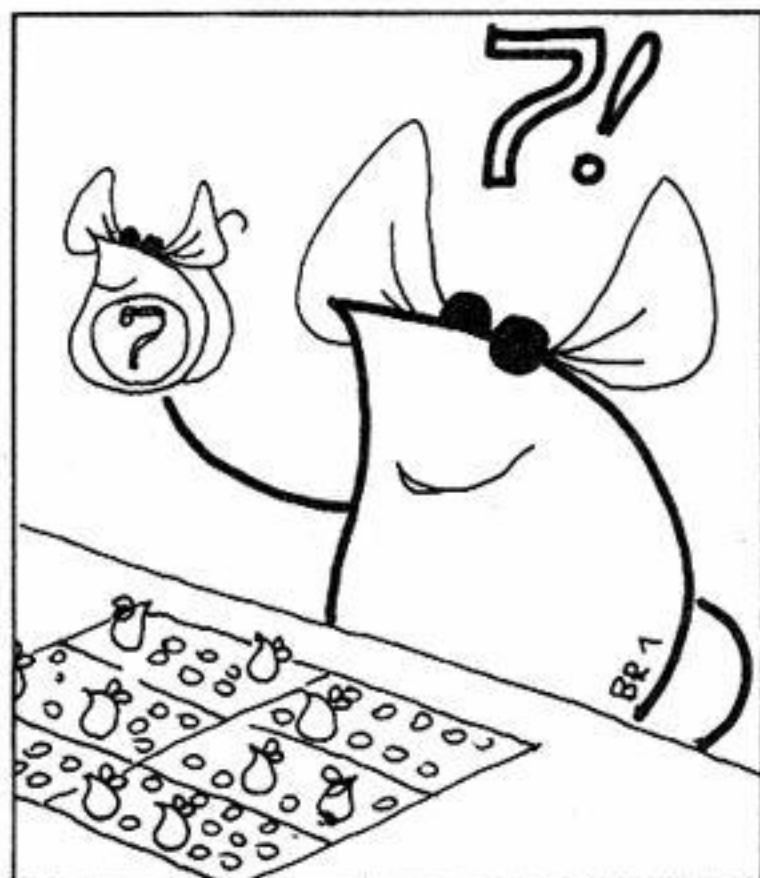
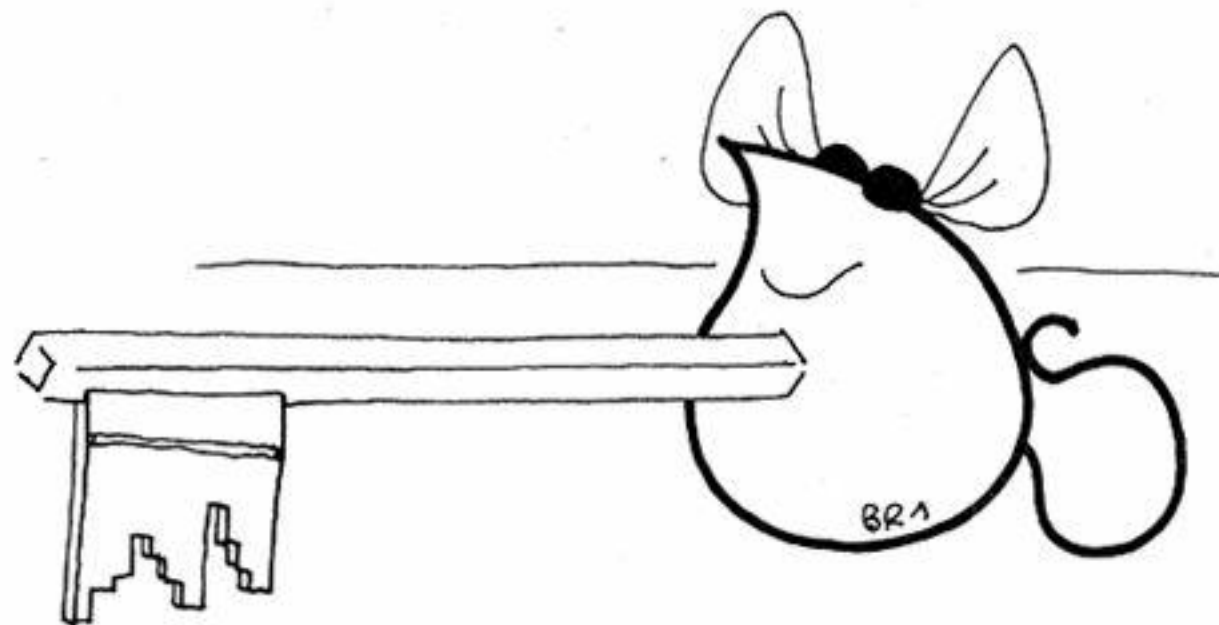
La parte più consistente ed interessante del programma, come sempre in giochi di questo genere, riguarda la **risposta** fornita da parte dell'elaboratore. Comunque le note presenti nel programma dovrebbero risultare di notevole aiuto a tutti coloro che fossero interessati ad uno studio della routine.

Le 6 carte in mano ai giocatori sono immagazzinate nella variabile **V** (numero giocatore, numero carta), mentre per controllare che non venga pescata col meccanismo casuale una carta già giocata, dopo ogni carta pescata, la variabile **Check** (valore, segno), viene posta ad 1.

Miglioratelo, ragazzi!

Il programma, come ogni listato che si rispetti, è suscettibile di variazioni ed ampliamenti di vario genere.

I più attenti osservatori si accorgeranno infatti che la risposta del computer alla mossa del giocatore è completamente guidata e nulla è lasciato alla casualità.

Solo nel caso in cui sia il computer a dover fare la prima mossa il meccanismo è, ovviamente, completamente casuale. Il consiglio è quindi quello di inserire una routine in grado di far "ragionare" il vostro elaboratore anche nel caso in cui debba giocare la prima carta.

```
seccar:
x = 1: o = 3 - o: IF ca = 40 THEN valore = valbri: segno = segbri
       IF ca > 39 THEN pesca ELSE GOSUB vase: GOTO pesca
prmossa:
ci = INT (3*RND)+1: IF v (2, ci) = 0 THEN prmossa
LOCATE 15, 15: PRINT v (2, ci);a$ (s (2, ci)): GOTO attesa
contr:
p1 = v (1, a)/10 + pu (v (1, a)): p2 = v (2, ci)/10 + pu (v (2, ci))
s1 = s (1, a): s2 = s (2, ci): x = 0: o = 1
  IF s1 = s2 AND p1 > p2 THEN o = 2: GOTO ok
  IF s1 = s2 AND p2 > p1 THEN ok
  IF s1 = segbri THEN o = 2
GOTO ok
vase:
valore = INT (10*RND)+1: segno = INT (4*RND)+1: IF check (valore, segno) = 1 THEN vase
check (valore, segno) = 1: RETURN
fine:
LOCATE 23, 1: IF p (2) > p (1) THEN PRINT "Hai vinto!": END
IF p (2) < p (1) THEN PRINT "Ho vinto!" ELSE PRINT "Pareggio"
END
```

```
100 REM Briscola. Versione Gw-Basic (tra parentesi quadre: modifiche per C/64)
110 REM by Dario Pistella 1991 per Commodore Computer Club
120 RITARDO = 10000: REM Loop per adattarsi alla velocita' del computer
130 REM Ritardo consigliato: 10000 per CPU 80386; 1000 per CPU 8088
140 RANDOMIZE (TIMER): A$(1) = "d": A$(2) = "b": A$(3) = "s": A$(4) = "c"
150 PU(1) = 11: PU(3) = 10: PU(10) = 4: PU(9) = 3: PU(8) = 2
160 CLS : FOR DA = 1 TO 3: FOR G = 1 TO 2: GOSUB 860
170 IF G = 1 THEN PRINT VL; A$(SG)
180 V(G, DA) = VL: S(G, DA) = SG
190 NEXT G, DA
200 GOSUB 860: LOCATE 20, 1: PRINT VL; A$(SG): REM briscola
210 VB = VL: SB = SG: CA = 6: GOSUB 590
220 REM mossa giocatore
230 A$ = ""
240 A$ = INKEY$: IF A$ < "1" OR A$ > "3" OR A$ = "" THEN GOTO 230
250 REM [C/64] GET a$: IF a$ < "1" OR a$ > "3" OR a$ = "" THEN GOTO 250
260 A = VAL(A$): IF V(1, A) = 0 THEN GOTO 230
270 LOCATE 15, 1: PRINT V(1, A); A$(S(1, A)): FOR RIT = 1 TO RITARDO: NEXT:
280 LOCATE A, 1: PRINT "        ": IF X THEN GOTO 790
290 REM *** computer decide se prendere o lasciare ***
```

```
300 PU = PU(V(1, A)): S = S(1, A): FOR Q = 1 TO 3
310 IF PU(V(2, Q)) > PU AND S = S(2, Q) AND PU(V(2, Q)) > 9 THEN CI = Q: O = 1: GOTO 620
320 NEXT: PU = PU + (V(1, A) / 10): IF PU < 3.9 THEN GOTO 420
330 REM prendi
340 RP = 1: FOR CI = 1 TO 3: Z = PU(V(2, CI)) + (V(2, CI) / 10)
350 IF S(2, CI) = S AND PU < Z THEN O = 1: GOTO 620
360 NEXT: IF S(1, A) = SB THEN GOTO 420
370 M = 30: MM = 0: FOR CI = 1 TO 3
380 IF S(2, CI) = SB AND M > V(2, CI) THEN M = V(2, CI): MM = CI
390 NEXT: IF M = 30 THEN GOTO 420
400 CI = MM: O = 1: GOTO 620
410 REM lascia
420 R = 30: U = R: UU = 0: RR = 0: FOR C = 1 TO 3: REM valori minori
430 VL = PU(V(2, C)) + (V(2, C) / 10): SC = S(2, C): IF VL = 0 THEN VL = 40
440 IF SC <> SB AND VL < U THEN U = VL: UU = C
450 IF SC = SB AND VL < R THEN R = VL: RR = C
460 NEXT
470 O = 2: IF S <> SB THEN 530
480 IF U <= 10 OR (U > 10 AND R >= 10) THEN CI = UU: GOTO 620
490 REM computer cede briscola
500 CI = RR: IF PU > R THEN GOTO 620
510 O = 1: GOTO 620
520 REM non briscola
530 IF U > 10 AND U < 30 AND S(2, UU) = S AND RP = 0 THEN GOTO 340
540 IF R < 30 AND ((U > 10 AND U < 30) OR U = 30) THEN GOTO 500
550 IF (U <= PU AND S(2, UU) = S) OR S <> S(2, UU) THEN CI = UU: GOTO 620
560 CI = UU: IF S = S(2, UU) THEN O = 1
570 GOTO 620
580 REM carte rimaste
590 LOCATE 21, 1: PRINT "rimaste"; : IF CA > 40 THEN PRINT 0:  ELSE PRINT 40 - CA
600 PRINT "tuoi:"; P(2), "computer:"; P(1): RETURN
610 REM ok
620 LOCATE 15, 15: PRINT V(2, CI); A$(S(2, CI)): FOR RIT = 1 TO RITARDO: NEXT:
630 P(O) = P(O) + PU(V(2, CI)) + PU(V(1, A)): CA = CA + 2: GOSUB 590
640 IF CA = 46 THEN GOTO 900:  ELSE VL = 0: SG = 0: IF CA < 41 THEN GOSUB 860
650 REM pesca
660 FOR K = 1 TO 1500: NEXT: LOCATE 15, 1: PRINT SPACE$(40)
670 IF O = 2 THEN V(1, A) = VL: S(1, A) = SG: LOCATE A, 1: PRINT VL; A$(SG)
680 IF O = 1 THEN V(2, CI) = VL: S(2, CI) = SG
690 IF X AND O = 1 THEN X = 0: GOTO 230
700 IF X THEN 760
710 REM seconda carta
720 X = 1: O = 3 - O: IF CA = 40 THEN VL = VB: SG = SB
730 IF CA > 39 THEN GOTO 660:  ELSE GOSUB 860: GOTO 660
740 REM prima mossa computer
750 CI = INT(3 * RND(1)) + 1: IF V(2, CI) = 0 THEN GOTO 750
760 REM [C/64] ci = INT(3 * RND(0)) + 1: IF v(2, ci) = 0 THEN GOTO 760
770 LOCATE 15, 15: PRINT V(2, CI); A$(S(2, CI)): GOTO 230
780 REM controllo
790 P1 = V(1, A) / 10 + PU(V(1, A)): P2 = V(2, CI) / 10 + PU(V(2, CI))
800 S1 = S(1, A): S2 = S(2, CI): X = 0: O = 1
810 IF S1 = S2 AND P1 > P2 THEN O = 2: GOTO 620
820 IF S1 = S2 AND P2 > P1 THEN GOTO 620
830 IF S1 = SB THEN O = 2
840 GOTO 620
850 REM valore e segno
860 REM [C/64] vl = INT(10 * RND(0)) + 1: sg = INT(4 * RND(0)) + 1: IF ck(vl, sg) = 1 THEN GOTO 860
870 VL = INT(10 * RND(1)) + 1: SG = INT(4 * RND(1)) + 1: IF CK(VL, SG) = 1 THEN GOTO 870
880 CK(VL, SG) = 1: RETURN
890 REM fine
900 LOCATE 23, 1: IF P(2) > P(1) THEN PRINT "hai vinto!": END
910 IF P(2) < P(1) THEN PRINT "ho vinto!":  ELSE PRINT "pareggio"
920 END
```

```
REM Briscola  (versione AMIGA)
REM by Dario Pistella
RANDOMIZE TIMER
a$ (1)  =  "d": a$ (2)  =  "b": a$ (3)  =  "s": a$ (4)  =  "c"
pu (1)  =  11: pu (3)  =  10: pu (10)  =  4: pu (9)  =  3: pu (8)  =  2
CLS: FOR dare = 1 TO 3: FOR g = 1 TO 2: GOSUB vase
  IF g  =  1 THEN PRINT valore;a$ (segno)
  v (g, dare)  =  valore: s (g, dare)  =  segno
NEXT g, dare
GOSUB vase: LOCATE 20, 1: PRINT valore;a$ (segno): REM briscola
valbri  =  valore: segbri  =  segno: ca  =  6: GOSUB carrim
attesa:
a$  =  INKEY$: IF a$  =  "" OR VAL (a$) < 1 OR VAL (a$) > 3 THEN attesa
a  =  VAL (a$): IF v (1, a)  =  0 THEN attesa
LOCATE 15, 1: PRINT v (1, a);a$ (s (1, a))
LOCATE a, 1: PRINT "    ": IF x THEN contr
REM  *** computer decide se prendere o lasciare ***
pu  =  pu (v (1, a)): s  =  s (1, a): FOR q = 1 TO 3
IF s  =  s (2, q) AND pu (v (2, q)) > 9 AND pu (v (2, q)) > pu THEN ci = q: o = 1: GOTO ok
NEXT: pu  =  pu +  (v (1, a)/10): IF pu < 3.9 THEN lascia
prendi:
rp = 1: FOR ci = 1 TO 3: z  =  pu (v (2, ci)) +  (v (2, ci)/10)
    IF s (2, ci)  =  s AND pu < z THEN o  =  1: GOTO ok
   NEXT: IF s (1, a)  =  segbri THEN lascia
m = 30: mm = 0: FOR ci = 1 TO 3
      IF s (2, ci)  =  segbri AND m > v (2, ci) THEN m  =  v (2, ci): mm  =  ci
     NEXT: IF m  =  30 THEN lascia
ci  =  mm: o  =  1: GOTO ok

lascia:
r  =  30: u  =  r: uu  =  0: rr  =  0: FOR c = 1 TO 3: REM valori minori
vl  =  pu (v (2, c)) +  (v (2, c)/10): sc  =  s (2, c): IF vl  =  0 THEN vl  =  40
  IF sc <> segbri AND vl < u THEN u  =  vl: uu  =  c
  IF sc  =  segbri AND vl < r THEN r  =  vl: rr  =  c
NEXT
o  =  2: IF s <> segbri THEN nonbri
   IF u < =  10 OR  (u > 10 AND r > =  10) THEN ci  =  uu: GOTO ok

dobri:
   ci  =  rr: IF r < pu THEN ok
o  =  1: GOTO ok

nonbri:
  IF u > 10 AND u < 30 AND s (2, uu)  =  s AND rp = 0 THEN prendi
  IF r<30 AND  ( (u > 10 AND u < 30) OR u  =  30) THEN dobri
  IF  (u < =  pu AND s (2, uu)  =  s) OR s <> s (2, uu) THEN ci  =  uu: GOTO ok
  ci  =  uu: IF s  =  s (2, uu) THEN o  =  1
GOTO ok

carrim:
  LOCATE 21, 1: PRINT "rimaste";: IF ca > 40 THEN PRINT 0 ELSE PRINT 40 - ca
  PRINT "tuoi: "p (2), "computer: "p (1): RETURN

ok:
LOCATE 15, 15: PRINT v (2, ci);a$ (s (2, ci))
p (o)  =  p (o) + pu (v (2, ci)) + pu (v (1, a)): ca  =  ca + 2: GOSUB carrim
IF ca = 46 THEN fine ELSE valore  =  0: segno  =  0: IF ca < 41 THEN GOSUB vase

pesca:
FOR k = 1 TO 1500: NEXT: LOCATE 15, 1: PRINT SPACE$ (40)
IF o  =  2 THEN v (1, a)  =  valore: s (1, a) = segno: LOCATE a, 1: PRINT valore;a$ (segno)
IF o  =  1 THEN v (2, ci)  =  valore: s (2, ci)  =  segno
IF x AND o  =  1 THEN x  =  0: GOTO attesa
IF x THEN prmossa
```

di Mauro Bossetti

Amiga ed i file relativi, gestiamoli con GfaBasic

AmigaBasic è un linguaggio interprete che lascia un po' a desiderare; per sostituirlo non c'è che l'imbarazzo della scelta. Stavolta diamo un'occhiata all'alternativa offerta da GfaBasic

Scopo di queste note è di illustrare il modo in cui il GfaBasic colloquia con il **drive**. Dato che, finora, abbiamo parlato di questo linguaggio in modo superficiale, sarà bene aprire una parentesi per descriverlo brevemente.

GfaBasic è un linguaggio estremamente veloce (quasi quanto il C) e possiede istruzioni molto sofisticate. Come altri moderni interpreti, è possibile ottenere copie compilate del programma, vale a dire che è possibile attivare una procedura (la **compilazione**, appunto) in grado di convertire, anche se parzialmente, il codice sorgente in linguaggio macchina, notoriamente veloce.

Con le seguenti righe, ad esempio...

```
FOR a = 1 to 100000
NEXT a
```

...in Amigabasic occorreranno **40.8** secondi per completare il ciclo, in GfaBasic solo **5.9**. Se, poi, si compila il programma, il tempo richiesto si riduce a **4.4**.

L'utilizzo ottimale del linguaggio si ha nella stesura di programmi... *intuitionizzati* con interfacce semplici da capire. La gestione della grafica, grazie alle potenti istruzioni, risulta estremamente veloce (da 3 a 4 volte in più rispetto all'AmigaBasic).

Nella nostra banca dati (che potete contattare usando un modem) troverete la conversione del programma Spirografo, pubblicato sul N. 77, in versione GfaBasic compilata, che funziona 3.5 volte più velocemente dell'Amigabasic. La gestione dei files sequenziali (argomento che, però, non viene qui preso in esame) è resa facile da istruzioni che permettono un agile spostamento all'interno del file stesso.

Le "chiamate" di libreria possono avvenire direttamente, poiché la **graphics** library, la **dos** library (e quasi tutte le altre) sono caricate automaticamente con l'interprete.

Il programma

Non viene presentata una parallela "traduzione" in Amigabasic in quanto il listato è semplicissimo da comprendere anche da chi non abbia mai visto in azione il GfaBasic (e, oltretutto, la sintassi sarebbe quasi identica).

In ogni caso è bene precisare che, a parte la gestione dei **files relativi** e dei **Bobs**, la sintassi ed i comandi dei due interpreti sono strepitosamente diversi. Nel circuito di pubblico dominio sono pre-

Non solo AmigaBasic

In tutte le confezioni di Amiga (modelli **500** e **2000**) è presente un dischetto (**Extras**) contenente il linguaggio interprete **AmigaBasic**.

Chi lo usa, però, non è soddisfatto a causa delle notevoli limitazioni che presenta sia in fase di editing, sia in fase di esecuzione, sia per la vetustà delle procedure possibili.

Da alcuni anni numerose software house propongono linguaggi interpreti alternativi, la cui sintassi rappresenta, ormai, un miscuglio (graditissimo dagli utenti) di **Basic**, **Pascal** e, talvolta, **C** ed **Assembly**.

GfaBasic è uno di questi linguaggi, molto usato da tutti coloro che, per un motivo o per l'altro, non si accontentano del linguaggio originale AmigaBasic (il quale, tra l'altro, vanta una paternità di tutto rispetto: **Microsoft**).

Pubblicheremo, di tanto in tanto, alcuni esempi che meglio possano orientare i lettori nella scelta di linguaggi interpreti alternativi.

Torneremo spesso, infatti, a trattare argomenti relativi al GfaBasic, appunto, ed all'altro ben noto interprete (**Amos**), altrettanto sofisticato e veloce, particolarmente vantaggioso nella gestione di musica e suoni.

COMMENTO	GFABASIC	A.BASIC
SCRITTURA		
Df0 (il file non esiste)	24.7	25.3
Df0 (il file esiste)	10.3	10.8
RAM	0.3	0.4
Fh1:	0.6	0.7
LETTURA		
Df0:	1.5	1.5
RAM	0.28	0.32

Tabella 1

La differenza di velocità non è elevata tra gli interpreti AmigaBasic e GfaBasic; tuttavia si deve pensare che eventuali efficienze di un linguaggio si scontrano inevitabilmente con la gestione intrinseca dell'hardware

Tipo	Simb.	Mem. occupata	Campo esist.
Boolean	!	1 byte (negli array 1 bit)	TRUE-FALSE
Byte	|	1 byte	0-255
Word	&	2 bytes	-32767-32767
Integer	%	4 bytes	-2147483648-2147483647
Float	# (opz.)	8 bytes	2.225073858507e+-308

Tabella 2

Tipi di variabili nel GfaBasic per Amiga

senti vari programmi di utilità in grado di operare le conversioni necessarie; si sottolinea che non sempre le conversioni ottenute sono perfettamente funzionanti, ma di certo provvedono a limitare, e di molto, il lavoro del programmatore.

Il mini-programma presentato, che non è di alcuna utilità pratica(!), riempie un file relativo di **50** record secondo una regola semplice da comprendere: i primi **49** record sono riempiti tutti dal carattere "**a**", l'**ultimo** di "**b**".

In seguito viene letto il file appena memorizzato ed effettuata la ricerca del carattere "**b**" costringendo il computer, in pratica, ad esplorare l'intero file nel tentativo di individuarlo.

Nella **tabella 1** sono riportati i tempi di esecuzione del corrispondente programma in Amigabasic (che si può ottenere, come già detto, con poche modifiche al listato); le sintassi dei due interpreti (nel caso specifico della gestione di un file relativo) sono molto simili tra loro. Hanno soprattutto il compito di far comprendere al lettore che il GfaBasic non richiede tempi eccessivi per il suo apprendimento, a patto che, ovviamente, si conosca abbastanza bene AmigaBasic.

Il file di nome **Prova** è stato salvato su un disco appena formattato (in due condizioni, riportate nella tabella 1), su una partizione da 17 megabyte piena al 7% di un **Hard Disk A2090** dotato di controller A2092 e nella Ram Disk (Chip Ram), utilizzando un normale(!) Amiga 2000 senza espansioni o schede velocizzatrici di alcun tipo. Nel test effettuato sono state eliminate le istruzioni di ricerca (**Instr**) e di stampa per velocizzare il procedimento. Si noti, innanzitutto, come il file venga aperto in modo identico. In GfaBasic in particolare, la prima lettera dopo **Open** (racchiusa tra apici) indica il tipo di file: **O** indica l'apertura in scrittura di un file sequenziale (che, se esiste già, viene cancellato); **I** apre un file sequenziale in lettura; **A** rende possibile l'aggiornamento di un file esistente (il puntatore viene automaticamente spostato alla fine del file); **U** apre un file esistente sia per la lettura che per la scrittura, mentre **R** si riferisce ai file relativi.

In quest'ultimo caso il valore numerico che segue il path del file aperto indica la *lunghezza* complessiva del record. E' importante far notare che un file può essere direzionato anche verso le periferiche (porta seriale, parallela, Ram, finestra Shell o Cli e stampante).

Procedendo, notiamo il comando **Field,** identico alla versione Amigabasic tranne per il fatto che se un "campo" contiene dati solamente numerici, lo si può dichiarare con **AT(x)** anziché con **AS stringa$**, sintassi che rende poi necessaria la conversione dal formato numerico a quello stringa.

Nel ciclo For Next si può osservare uno strano simbolo dopo la variabile d: è la barra verticale |. Il GfaBasic permette la gestione di ben **6** tipi differenti di variabili come indicato in **tabella 2.** Non spaventatevi: potete continuare ad usare le tradizionali variabili in virgola mobile che non richiedono simboli particolari ed evitano calcoli dovuti al campo di esistenza delle variabili.

Comunque, più una variabile occupa memoria, più è lenta da gestire. Per quanto riguarda If ed Endif, si vede chiaramente che il Then è superfluo. Ma come mai si usa Endif anche se vi è una sola istruzione dopo l'If? E qui entra in gioco **l'editor del GfaBasic**, che **non permette di avere più di una sola istruzione per linea**.

Se ciò può sembrarvi scomodo, cambiate subito idea: i listati scritti in questo modo sono estremamente leggibili, anche perché l'editor "indenta" automaticamente tutte le istruzioni comprese tra i cicli. GfaBasic ha la sua forza nella possibilità di stesura di programmi estremamente strutturati, fattore che facilita un eventuale apprendimento del Turbo Pascal o del C (dove la strutturazione è obbligatoria).

Il famigerato **GoTo** non è stato eliminato del tutto, ma grazie alle numerosissime istruzioni specifiche per i cicli (**If** ... Endif, For ... Next, Repeat ... Until, While ... Wend, e la flessibile Do ... Loop) diventa inutile.

A proposito dell'editor, è bene precisare che questo controlla la corretta sintassi delle istruzioni **in tempo reale** dal momento che **non** permette di lasciare una linea che contiene un errore di sintassi; verifica, addirittura, il numero dei parametri passati e tutte le istruzioni che tirano in ballo le librerie.

La nostra breve chiacchierata introduttiva ha qui termine. Ci addentreremo in futuro in argomenti più complessi, che permetteranno però di evidenziare l'estrema potenza (e velocità) del linguaggio. Per ora l'importante è incominciare.

Magari procurandosi il GfaBasic...

```
' SPIROGRAFO GfaBasic Amiga
' (per la versione AmigaBasic
' rileggere C.C.C. n.77)
main:
OPENW #0
~ActivateWindow(WINDOW(0))
TITLEW #0,"Autore Roberto
Morassi....Conversione di
Mauro Bossetti"
DEFFN
x(x)=320+(r*COS(x+d)+z*COS(-x*t1
+d))*1.9
DEFFN
y(x)=100-(r*SIN(x+d)+z*SIN(-x*t1
+d))
p=3.1415
PRINT "Esempi di risposte
Tempi:    GFABASIC compilato
Amigabasic"
PRINT "n,90,50,0.3,1.05,f
6.8                        24.5"
PRINT "n,120,28,0.5,1.01,f
9.5                        35.2"
PRINT "s,90,200,12,0.8,1.1,f
4.1                        15.2"
PRINT "n,10,7,0.5,1,f
9.4                        34.1"
PRINT "n,137,100,0.6,1,f
137.2                       504.8"
PRINT "s,90,351,167,0.5,1.1,v
3.5                        12.6"
PRINT "s,360,100,77,0.7,1.02,f
106.7                       378.5"
PRINT "s,360,100,77,0.7,1,v
3.5                        12.7"
INPUT "Rotazione asse (s/n)";a$
IF a$<>"s"
  d=p/2
  GOTO ruotafissa
ENDIF
INPUT "Angolo di rotazione";d
d=p/2-d*p/180
ruotafissa:
INPUT "n. denti ruota
fissa";rf&
INPUT "n. denti ruota
mobile";rm&
t=rf&/rm&
t1=t-1
IF t<=1
  GOTO ruotafissa
ENDIF
decentramento:
INPUT "Decentramento";dc
z=dc*90/t
INPUT "Ampiezza linea";amp
IF amp=0
  amp=1.1
ENDIF
mx=rf&
FOR x=1 TO rf&
  IF x*rf&/rm&=INT(x*rf&/rm&)
    mx=x*rf&/rm&
    x=rf&
  ENDIF
NEXT x
mx=2*PI*mx/t
r=90*(1-1/t)
INPUT "Incremento fisso o
variabile (f/v)";a$
v=mx/400
IF a$="f"
  v=PI/80
ENDIF
INPUT "Bitplanes (1-4)";ccol|
IF ccol|>2
  CLOSEW #0
  OPENS
1,0,0,640,256,ccol|,32768
  OPENW #0
ELSE
  CLEARW #0
ENDIF
SELECT ccol|
CASE 1
  tt|=1
CASE 2
  tt|=3
CASE 3
  tt|=7
CASE 4
  tt|=15
ENDSELECT
col|=1
x1=(320+(z+r)*COS(d))
y1=100-((z+r)*SIN(d))
iniz=TIMER
FOR a=0 TO mx STEP v
  x=FN x(a)
  y=FN y(a)
  COLOR col|
  LINE x1*amp,y1*amp,x,y
  x1=x
  y1=y
  IF col|<tt|
    INC col|
  ELSE
    col|=1
  ENDIF
NEXT a
fine=TIMER
PRINT "Secondi";(fine-iniz)/200
INPUT "Ancora";a$
IF a$="s"
  CLEARW #0
  LOCATE 1,1
  GOTO main
ENDIF
```

Spirografo in GfaBasic

Per esigenze di impaginazione alcune righe sembrano essere suddivise in più righi fisici. E' ovvio, invece, che bisogna premere il tasto Return solo dopo aver digitato l'ultimo parametro delle varie istruzioni

```
OPEN "r",#1,"ram:Prova",200
FIELD #1,200 AS c$
FOR d|=1 TO 200
  a$=a$+"a"
  b$=b$+"b"
NEXT d|
c$=a$
t=TIMER
FOR d|=1 TO 49
  PUT #1,d|
NEXT d|
c$=b$
PUT #1,50
t1=TIMER
PRINT (t1-t)/200
t=TIMER

FOR d|=1 TO 50
  GET #1,d|
  e=INSTR(c$,"b")
  IF e<>0
    PRINT "Caratt. b in posiz.";e
  ENDIF
NEXT d|
t1=TIMER
PRINT (t1-t)/200
```

GfaBasic e file relativi

Si noti l'estrema semplicità richiesta dalla sintassi dell'interprete

A che punto siamo? (indagine tra i lettori)

Come nostra abitudine (anzi, siamo un po' in ritardo rispetto al passato...) proponiamo, ai nostri lettori, di esprimere il proprio giudizio sulla nostra rivista: tutto ciò che volete comunicarci, quantomeno le opinioni più importanti, potete farcele conoscere compilando il semplice questionario di queste due pagine; Non abbiate timore a staccare il foglio: seguendo la linea tratteggiata eviterete che la pagina "opposta" si distacchi dal fascicolo. Chi lo desidera, ovviamente, potrà inviare la fotocopia riproducente l'inchiesta.

Domanda N. 1 (esperienze precedenti)

Da quale esperienza informatica provieni?

- [x] Autodidatta (hobby, club, amici, riviste, ecc.)
- [ ] Ho seguito un corso (scuola, università, corrispond.)
- [ ] Lavoro nel settore specifico

Dom.N. 2 (computer usati nel passato)

Quali computer hai usato in precedenza?

- [x] Il C/64 (oppure Vic 20, C/128, C/16, +4, Pet)
- [ ] Lo Spectrum (oppure Msx, QL, vecchi Apple)
- [ ] Amiga oppure Ms - Dos

Dom.N. 3 (computer usati OGGI)

Quali computer usi abitualmente?

- [ ] Quello/i indicato/i nella domanda N.2
- [x] Amiga
- [ ] Ms - Dos

Dom.N. 4 (computer che userai)

Quale computer pensi di usare nel corso del 1992?

- [ ] Quello/i indicato/i nelle dom. 2 e/o 3
- [x] Amiga
- [ ] Ms - Dos

Dati del lettore (compilazione facoltativa)

Cognome

Nome

Indirizzo

(CAP) Città **Tel.**

Domanda N. 5 (configurazione)

Quanti drive ha il sistema indicato nella dom. n. 4?

❑ 1 da 5.25 ❑ 2 (o più) da 5.25

❑ 1 da 3.5 ❑ 2 (o più) da 3.5

❑ Hard disk Megab. (specificare)

Domanda N. 6 (memoria RAM)

Di quanta RAM dispone il sistema (ind. in doman. 4)?

❑ 500 K(se Amiga) oppure 640 K(se Ms - Dos)

❑ 1 mega

❑ N. megabyte (specificare):

Domanda N. 7 (stampante)

Quale stampante userai con il sistema di dom. 4?

❑ Un modello a 9 aghi (oppure nessuna)

❑ Un modello a 24 aghi (o comunque di medio prezzo)

❑ Una laser (o un modello di prezzo elevato)

Domanda N. 8 (modem)

Nel 1992 disporrai di un modem?

❑ No

❑ Sì, un modello medio (prezzo fino a 400 mila lire)

❑ Sì, un modello di elevate prestazioni (oltre le 500 mila)

Dom. N. 9 (prezzo della rivista)

Saresti disposto all'aumento del prezzo di copertina se decidessimo di inserire un dischetto (ovviamente del formato valido per il computer indic. in dom. 4)?

❑ No, in nessun caso il prezzo deve aumentare

❑ Sì, purchè contenga *anche* giochi e simili

❑ Sì, purchè contenga *anche* s/w di tipo professionale

❑ Sì, purchè contenga *anche* utility

❑ Sì, purchè il prezzo di copertina non risulti mai superiore a L. (indicare cifra max.)

N.B: è possibile barrare più risposte, purchè non in contraddizione tra di loro

Inserire in busta chiusa, affrancare e spedire a:

Systems Editoriale
Via Mosè 22
cap 20090 Opera (Mi)

Dom.N. 10 (Argomenti preferiti)

A quali argomenti vorresti si dedicasse più spazio?

❑ Recensioni di software e/o hardware

❑ Didattica (linguaggi C, Basic, Pascal, Assembly, ecc.)

❑ Uso ottimale di programmi ed utility commercializzati (Word processor, DTP, Spreadsheet, Database, ecc.)

Un'importante precisazione

Prima di rispondere alle domande indicate, ricorda che molte di queste si riferiscono al sistema computerizzato indicato nella **domanda N.4** (quello, cioè, che ritieni di utilizzare in prevalenza nel corso del prossimo anno **1992**). Se ritieni che continuerai ad usare il computer che usi tuttora (segnalato nella domanda **n. 3**), ovviamente, le risposte che fornirai dovranno riferirsi a quest'ultimo.

di Ugo Spezza

Tre programmi risolvono le equazioni di terzo grado

Le equazioni: prima o poi qualsiasi studente ha a che fare con esse, o le ha già incontrate nel corso dei suoi studi.

Le equazioni, in genere, sono tutt'altro che semplici entità astratte generate dalla mente schizoide dei matematici; la maggior parte (per non dire la totalità) dei problemi di matematica, fisica, elettrotecnica, meccanica, eccetera si risolve impostando la relativa equazione.

Le prime equazioni che si incontrano negli studi sono quelle di primo grado, del tipo $ax + b = 0$, la cui soluzione è talmente banale che non vale la pena di parlarne.

Vi sono, poi, quelle di secondo grado (del tipo $ax^2 + bx + c = 0$) che, come è noto si risolvono mediante...

X = (-b +- Sqr (D)) / 2a

...in cui $D = b^2 - 4ac$ è definito come "discriminante". Le radici reali **X1** e **X2** possono risultare reali **distinte**, **coincidenti** (X1 = X2) o **complesse coniugate** X1 +- jX2.

Il microprogramma di queste pagine (utile per chi ha comprato mezz'ora fa il suo primo computer...) risolve un sistema di equazioni di secondo grado.

I "veri" problemi, però, cominciano quando si studiano le equazioni di **terzo grado**.

La ricerca di una formula in grado di estrarre le radici di un'equazione di grado superiore al secondo (di cui si conoscessero i soli coefficienti ed il termine noto) fu uno dei problemi affrontati da un'intera generazione di matematici, tra cui si distinse l'italiano **Tartaglia**.

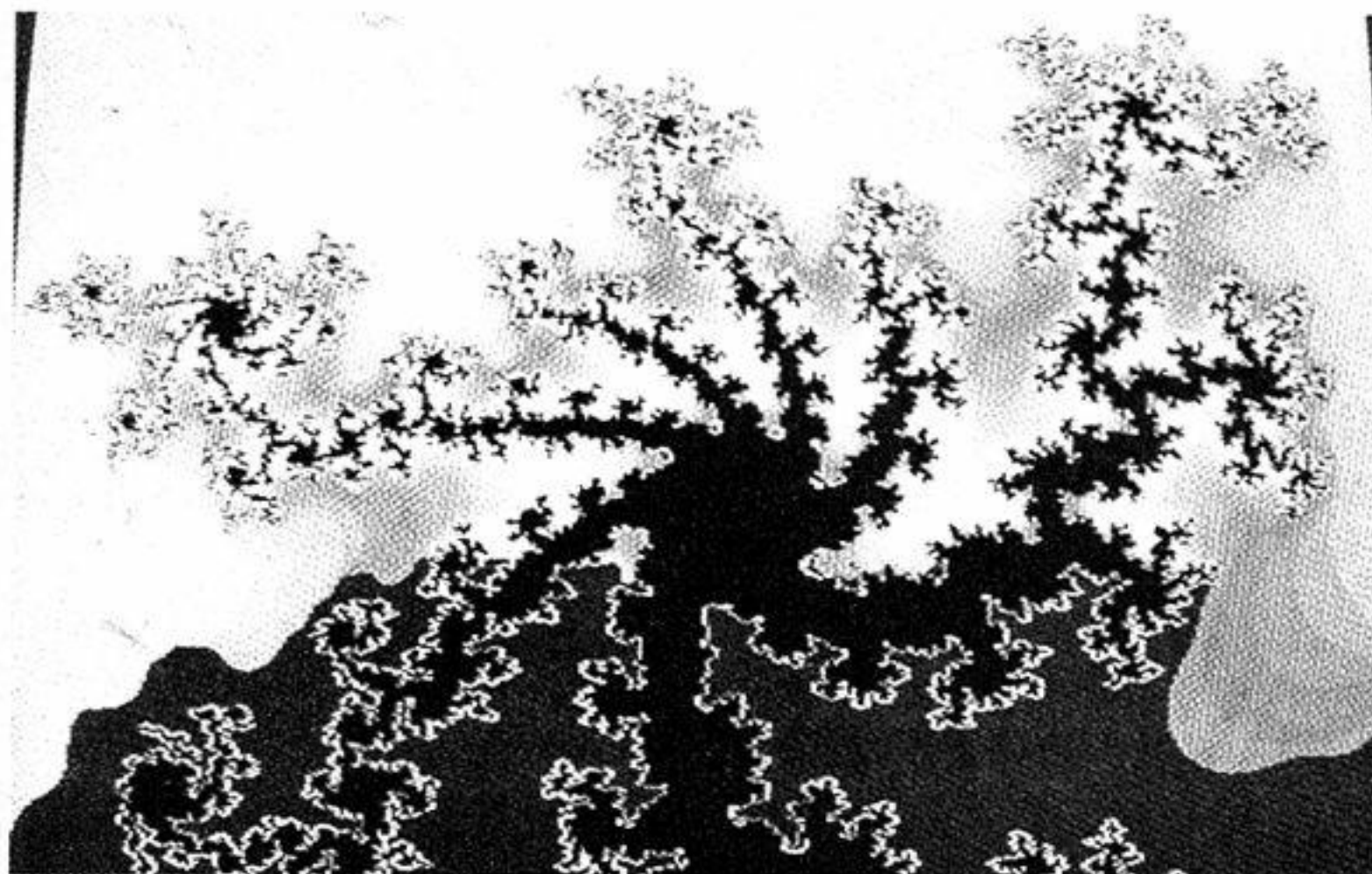
Il metodo insegnato nelle scuole per risolvere il problema si basa sulla regola di **Ruffini**, che però funziona solo se si riesce a trovare (a tentoni) almeno una delle radici dell'equazione. Questo è un limite inaccettabile, tanto più che la matematica, come è noto, tutto è fuorché una scienza empirica.

Un metodo che permette di determinare con precisione le radici dell'equazione è il metodo di **Cardano**, che è tanto complicato da scoraggiare una sua applicazione "manuale".

Se il computer lavora

Un algoritmo, per quanto sia complesso, può esser sviluppato da un qualsiasi computer, pur se programmato in Basic.

Avrete capito che dovete limitarvi a trascrivere uno dei due listati Basic di queste pagine; potete scegliere tra la versione **AmigaBasic** oppure **Gw Basic**. Il primo dei due è un programma davvero completo, mentre il secondo, ridotto all'osso, può essere digitato in Gw Basic, oppure in AmigaBasic e persino sui vecchi **C/64 - 128**.

Nel listato AmigaBasic potete digitare i simboli di elevazione a potenza (al quadrato e al cubo) premendo **Alt + 2** oppure

Alt + 3. Con Amiga, infatti, la particolare combinazione di tasti (**Alt** + **1**, **2**, **3**) riproduce, rispettivamente, i tre valori leggermente spostati in alto *(esponenti)*. Il programma rappresenta, inoltre, un'occasione unica per imparare l'uso delle funzioni stringa in Basic e per visualizzare un messaggio superando le tipiche limitazioni del comando Print.

Il programma

Per spiegare integralmente il metodo di Cardano non basterebbe l'intero fascicolo; si può tuttavia affermare che l'algoritmo si basa sul presupposto che i coefficienti dei termini della equazione siano diversi da zero.

Se, infatti, uno solo di essi fosse eguale a zero, l'equazione sarebbe, in realtà, di secondo grado e, come tale, risolvibile con un programma meno complesso.

Nel metodo usato, tutti i termini vengono divisi per il coefficiente del termine di terzo grado. Supponendo che il termine di secondo grado si annulli per un certo valore di X, si riesce a ridurre l'equazione in forma normale e le soluzioni vengono valutate in base al valore assunto dal discriminante.

Viene dapprima definita la variabile **ER** = **1** / **1000000**; quindi tutti i valori che differiscono per meno di **ER** sono da considerare come soluzioni coincidenti; i valori vengono testati dalla variabile **C**.

Un'equazione di terzo grado presenta sempre tre soluzioni, inquadrabili in vari casi: tutte e tre reali; una reale e due complesse coniugate; una reale e due coincidenti. Il metodo di Cardano è in grado di risolvere qualsiasi equazione di terzo grado e, se si pensa che è stato ideato quando i computer non esistevano, ci si può rendere conto della grandezza del suo ideatore.

Come gira il programma

Dopo il Run digitate, in risposta alle domande sul valore dei coefficienti di X^3, X^2, X ed il termine noto, rispettivamente:

2, 5, -2, -5

Se non avete commesso errori, dovranno risultare le radici X1 = **1** [R], X2 = **-2.5** [R], X3 = **-1** [R], ove **[R]** significa che il risultato è Reale, **[I]** Immaginario.

Con *a = 1, b = 1, c = -2, d = -2*, le radici saranno X1 = Sqr (2) [R] , X2 = -SQR(2) [R], X3 = -1 [R].

Ancora, con *a = 1, b = 1, c = 1, d = 1*, le radici saranno X1 = -1 [R], X2 = 0 [I], X3 = 1 [I]. I risultati saranno di pochissimo approssimati, cosa che potete evitare usando Print Using in luogo di Print.

Infine, con *a = 1, b = 1, c = -2, d = -1*, le radici saranno X1 = 1.24698 [R], X2 = -1.801938 [R], X3 = -0.4450419 [R].

```
' Matematica per AmigaBasic by
' by Ugo Spezza 1991
'  EQUAZIONI DI TERZO GRADO
SCREEN 1, 640, 256, 2, 2: WINDOW 2, "", , 0, 1
CLS: PALETTE 0, 0, 0, 0: PALETTE 1, 0, .8, 0
PALETTE 2, 0, 1, 1: COLOR 2: q$ = CHR$ (13)
PRINT q$ + "  SOLUZIONE DI EQUAZIONI DI TERZO GRADO" + q$ + q$
INPUT "  Coefficiente di x3 "; d
PRINT : INPUT "  Coefficiente di x2 "; f
PRINT : INPUT "  Coefficiente di x "; g
PRINT : INPUT "  Termine noto "; h
w = SGN (f) + SGN (g) + SGN (h): IF w = 1 THEN w = 2
IF w = -1 THEN w = 1
CLS: LOCATE 4, 3
PRINT "FORMA ALGEBRICA DELLA EQUAZIONE:  ";
ra$ = STR$ (d) + "x3" + STR$ (f) + "x2" + STR$ (g) + "x" + STR$ (h)
FOR q = 1 TO w
sp = INSTR (2, ra$, " ")
MID$ (ra$, sp) = " + "
NEXT
PRINT ra$ + " = 0"
GOSUB routinecalcolo
LOCATE 10, 10: PRINT "SOLUZIONI DELLA EQUAZIONE: " + q$ + q$
COLOR 3: PRINT "   X1 = "; x1, x1$ + q$: PRINT "   X2 = "; x2, x2$ + q$
PRINT "   X3 = "; x3, x3$: LOCATE 25, 3
PRINT"[ENTER] PER CONTINUARE" + q$ + q$ + "  [ESC] PER USCIRE"
LINE  (10, 60)- (340, 150), 2, b
1 a$ = INKEY$: IF a$ = CHR$ (27) THEN SCREEN CLOSE 1: END
IF a$<>q$ THEN 1
RUN
```

Prima parte del listato AmigaBasic

```
routinecalcolo:
e = .0000001: pi = 3.14159265359#
f = f/d: g = g/d: h = h/d
f = f/3: d =  (g/3)- (f^2)
k = h- (f*g) +  (2*f^3)
c =  (4*d^3) +  (k^2)
IF e>ABS (c) THEN coincidente
IF c>0 THEN nonreale
a = 2*SQR (-d)
x = k/ (2*d*SQR (-d))
d = pi/2
b = -ATN (x/SQR (-x*x + 1)) + d
k = ATN (.5/SQR (.75))
g = a*SIN (d- (b/3)): h = -a*SIN (k +  (b/3)): i = -a*SIN (k- (b/3))
g = g-f: h = h-f: i = i-f
x1 = g: x2 = h: x3 = i
x1$ = "Reale": x2$ = x1$: x3$ = x1$
RETURN
nonreale:
c = SQR (c): a =  (c-k)/2: b = - (c + k)/2: c = 1/3
a = ABS (a)^c*SGN (a)
b = ABS (b)^c*SGN (b): c = SQR (3)/2
x1 = a + b-f: x1$ = "Reale"
x2 = -.5* (a + b)-f: x2$ = "Immaginario"
x3 = c*ABS (a-b): x3$ = x2$
RETURN
coincidente:
IF e>ABS (d) THEN x1 = -f: x2 = -f: x3 = -f: x1$ = " ": x2$ = " ": x3$ = " "
RETURN
a = -ABS ( (k/2)^ (1/3))*SGN (k)
x1 = 2*a-f: x2 = -a-f: x3 = x2
x1$ = "reale": x2$ = "Coincidente": x3$ = x2$
RETURN
```

Seconda parte del lsitato AmigaBasic

La maggior parte delle equazioni di terzo grado sono simili alle ultime due, e quindi non risolvibili con la regola di Ruffini.

Le stringhe in Basic

Come già detto, il programma **AmigaBasic** è anche un ottimo banco di studio per coloro che vogliono imparare la gestione del testo e delle stringhe.

All'inizio vengono definiti i colori con l'istruzione **Palette**. Se tale "preziosismo" può apparire inutile, pensate che, nel caso in cui (con **Preferences**) siano stati alterati i colori base, il testo o alcune parti del programma potrebbero non esser più visibili.

Ma veniamo alle stringhe: vedremo come, usando il solo comando Print, visualizzare una qualsiasi formula algebrica in un modo del tutto identico a quello di un qualsiasi testo di matematica, rispettando cioè il posizionamento degli esponenti.

La funzione **Str$(n)** (in cui **n** è un valore numerico), fornisce una stringa uguale al numero; ad esempio, ponendo **A$ = Str$(54)**, ed eseguendo **Print A$**, sarà visualizzata la stringa **54** poiché, appunto, **A$ = "54"**.

Inserendo opportunamente i caratteri visualizzabili con la pressione dei tasti **Alt** + **2** e **Alt** + **3**, pertanto, si può ottenere la stringa **ra$** che, alla fine, risulterà formata dai simboli x^3, x^2, x e da altri valori numerici qualsiasi.

C'è da risolvere il problema che il comando Print presenta quando stampa i numeri positivi; questi vengono visualizzati preceduti da uno spazio e non dal simbolo più (**+**).

Lo spazio vuoto, però, altro non è che un carattere Ascii (ed esattamente quello cui corrisponde il codice **32**) e come tale può essere trattato. Dal momento che siamo in presenza di un'equazione, ed i termini positivi possono essere al massimo tre, altrettante possono essere le

```
10 REM Equaz. II grado
20 INPUT "COEFFICIENTI A, B, C "; A, B, C
30 D = B^2 - 4*A*C: PRINT
40 IF D<0 THEN PRINT "RADICI IMMAGINARIE: ";-B/2/A;"+-j";(-D)^(1/4):END
50 IF D=0 THEN PRINT"RADICE UNICA: ";-B/2*A:END
60 PRINT "RADICI: ";(-B-SQR(D))/2*A,(-B+SQR(D))/2*A
```

```
5  REM   (EQUAZIONI DI TERZO GRADO)
10 REM  Matematica per Computer Ms - Dos
20 REM  Ugo Spezza 1991
30 REM
40 CLS: INPUT "Coefficiente di x3 "; d
50 PRINT : INPUT "Coefficiente di x2 "; f
60 PRINT : INPUT "Coefficiente di x "; g
60 PRINT : INPUT "Termine noto "; h
70 q$ = CHR$ (13): GOSUB 100
80 PRINT q$; q$; "SOLUZIONI DELLA EQUAZIONE: "; q$; q$
90 PRINT "X1 = "; x1, x1$; q$: PRINT "X2 = "; x2, x2$; q$
95 PRINT "X3 = "; x3, x3$: END
100 e = .0000001: pi = 3.14159265359#
110 f = f/d: g = g/d: h = h/d
120 f = f/3: d =  (g/3)- (f^2)
130 k = h- (f*g)+ (2*f^3)
140 c =  (4*d^3)+ (k^2)
150 IF e>ABS (c) THEN 340
160 IF c>0 THEN 270
170 a = 2*SQR (-d)
180 x = k/ (2*d*SQR (-d))
190 d = pi/2
200 b = -ATN (x/SQR (-x*x+1))+d
210 k = ATN (.5/SQR (.75))
220 g = a*SIN (d- (b/3)): h = -a*SIN (k+ (b/3)): i = -a*SIN (k- (b/3))
230 g = g-f: h = h-f: i = i-f
240 x1 = g: x2 = h: x3 = i
250 x1$ = "Reale": x2$ = x1$: x3$ = x1$
260 RETURN
270 c = SQR (c): a =  (c-k)/2: b = - (c+k)/2: c = 1/3
280 a = ABS (a)^c*SGN (a)
290 b = ABS (b)^c*SGN (b): c = SQR (3)/2
300 x1 = a+b-f: x1$ = "Reale"
310 x2 = -.5* (a+b)-f: x2$ = "Immaginario"
320 x3 = c*ABS (a-b): x3$ = x2$
330 RETURN
340 IF e>ABS (d) THEN x1 = -f: x2 = -f: x3 = -f: x1$ = " ": x2$ = " ": x3$ = " "
350 RETURN
360 a = -ABS ( (k/2)^ (1/3))*SGN (k)
370 x1 = 2*a-f: x2 = -a-f: x3 = x2
380 x1$ = "reale": x2$ = "Coincidente": x3$ = x2$
390 RETURN
```

La versione Gw - Basic

stringhe spazio da sostituire. Occorre allora impostare una ricerca delle stringhe spazio servendosi della funzione **N = Instr (x, A$, B$)** ove **x** è il punto di partenza della ricerca della stringa **A$** all'interno della stringa **B$** (di maggiore dimensione).

Nel nostro caso **x = 2** dal momento che possiamo fare a meno del segno del coefficiente di x^3. Si imposta quindi un ciclo For... Next, vengono contati i numeri positivi mediante Sgn(x), e una volta trovata la stringa spazio, questa viene sostituita con la stringa "**+**" usando la funzione Basic **MID$ (b$, n) = C$**.

Nel nostro caso *[MID$ (ra$, sp) = "+"]* **b$** è la stringa da esaminare, **c$** è la stringa sostitutiva ed **n** è il punto di partenza della sostituzione, numero ottenuto da **Instr()**, come già detto.

Con il semplice linguaggio Basic, pertanto, si possono ottenere formattazioni di testi di qualunque tipo, alla pari di complessi linguaggi ad alto livello (C, Pascal...). Potete addirittura creare caratteri personalizzati con la utility **FED** (disco Extras) e caricarli nel sistema con **FF** (directory C del Workbench), ma vi occorrerà molta pazienza.

Per esercizio, provate ad eliminare le stringhe "**1**" che vengono a crearsi in talune espressioni; difatti, anche se la forma $1x^2$ è algebricamente corretta, sui testi si usa omettere l'"1" riferendosi ad un più sbrigativo x^2.

POSTAMIGA

(a cura di Domenico Pavone)

Un salto qua ed uno là

Vorrei sapere quali sono le differenze tra le istruzioni Assembler Bsr e Jsr, visto che servono entrambe per saltare ad un sottoprogramma.

(M. Mattei - S.Benedetto V.S.)

Da un punto di vista formale, le due istruzioni svolgono effettivamente lo stesso lavoro: depositano sullo **Stack** l'indirizzo di ritorno, eseguono la subroutine specificata dall'operando, quindi "ripescano" l'indirizzo lasciato sullo Stack e tornano ad eseguire l'istruzione che segue immediatamente quella di salto.

La differenza sta nel fatto che **Bsr** (**B**ranch To **Subrou**tine) opera un salto **relativo**, mentre **Jsr** (**J**ump To **Subrou**tine) provoca un salto **assoluto**.

Detto più semplicemente, Jsr funziona in pratica come Jmp (a parte il ritorno), ovvero opera un Jump incondizionato alla locazione di memoria specificata, che potrà così essere una qualunque di tutto l'ambito indirizzabile dal Microprocessore (per il 68000 = 16 Mb). Il che significa, in termini pratici, che non sarà necessario preoccuparsi di nulla.

Bsr, invece, è più simile alle istruzioni di salto condizionato, nel senso che sfrutta uno "scostameto" (**offset**) dal suo indirizzo corrente per stabilire l'indirizzo di arrivo.

Banalmente: si supponga che all'indirizzo **X** si trovi una istruzione `Bsr Miaroutine`. Generalizzando, l'istruzione Bsr funziona come se calcolasse la differenza tra la locazione della label Miaroutine e dell'indirizzo X (differenza che chiameremo Y), e di fatto dicesse: salta di Y locazioni (in avanti o all'indietro) a partire da quella attuale.

L'effetto risulterebbe uguale a quello ottenibile con Jsr, ed anzi leggermente più veloce (**10** cicli macchina contro **14**), ma con un limite ben preciso: poiché per memorizzare lo scostamento è disponibile al massimo una word di **16 bit** (2 byte), questo non può superare i **32 Kb** in avanti o all'indietro.

Un limite che può sembrare tutto sommato molto ampio, ma non si dimentichi che, se per esempio viene richiamata una funzione di libreria, non si può sapere a priori dove questa verrà allocata.

Di conseguenza, Bsr può essere adoperato soprattutto per salti nell'ambito di routines la cui brevità sia garantita, mentre per tutti gli altri casi va sfruttata Jsr.

Considerato poi il guadagno davvero minimo in termini di velocità, il consiglio è di ricorrere *sempre* a Jsr, almeno fino a che non si è del tutto padroni del sistema operativo di Amiga.

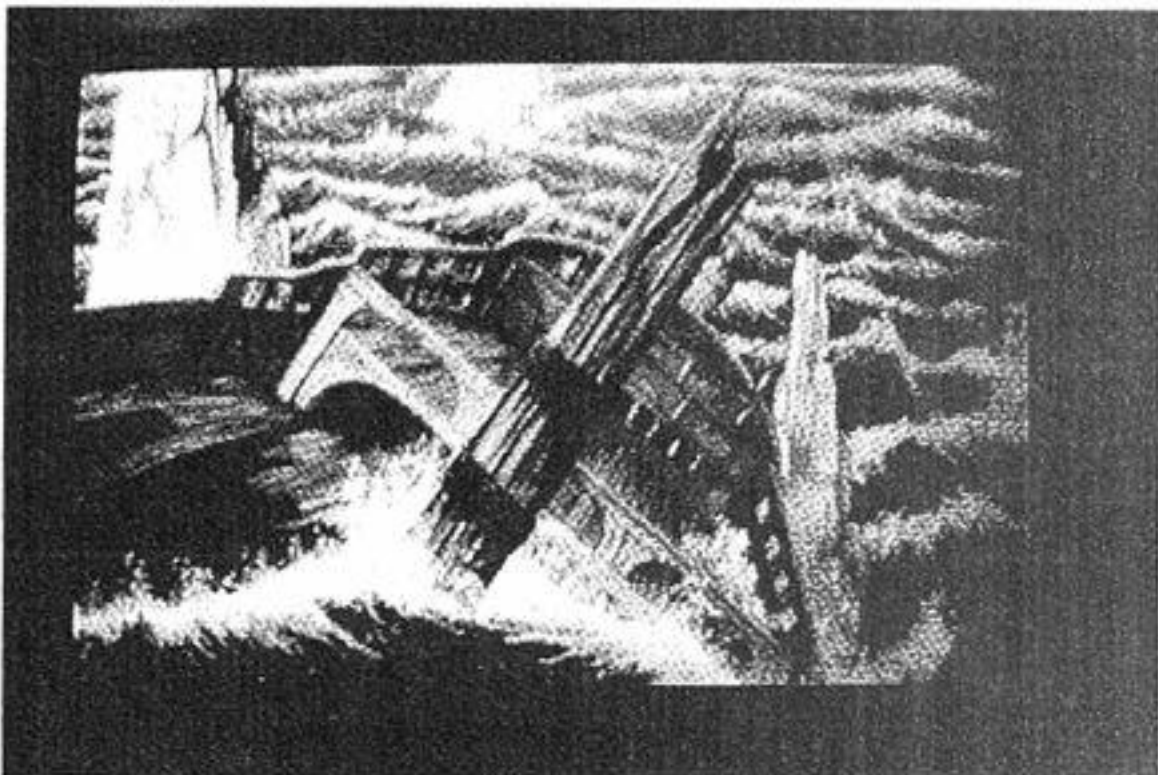
Ammesso che si possa mai diventarlo...

Un Ask migliore

Adoperando Ask in un batch file per richiedere un input, si può fare in modo che accetti qualcosa di diverso da "Y" e "N"?

(Sergio Licastro - Palermo)

Con il comando incluso nel disco Workbench, no.

Uniche alternative: riscrivere da sé il comando, o ricorrere ad **Arp**, il completo sistema di Pubblico Dominio che sostituisce con efficienza decisamente maggiore molti dei normali comandi del Dos (ma non solo).

Il comando Ask implementato da Arp, infatti, consente di definire liberamente i due termini accettabili in input, con la possibilità addirittura di non limitarli ad un solo carattere. Giusto per fare un esempio, adoperando la forma...

```
Ask "Rispondi" "S" "N"
```

...verrebbe stampata la stringa *"Rispondi"*, dopodiché immettendo **S** verrebbe settata la condizione **Warn** come di consueto, mentre **N** risponderebbe a **Not Warn**. Piuttosto che *S* e *N* potrebbero anche essere adoperati **Si** e **No**, nel qual caso andrebbero obbligatoriamente digitati in input per esteso. A differenza dell'Ask di Amigados, si avrebbe inoltre il vantaggio che, in caso di errata digitazione dell'input, questo verrebbe riproposto con la segnalazione delle due scelte possibili.

Per adoperare Ask oppure tutti i comandi di Arp, è sufficiente procurarsi il pacchetto software dalle fonti più comuni del pubblico dominio (soprattutto le varie **bbs**), e copiare nella directory **Libs** del disco **Workbench** il file **Arp.library**. Così facendo, risulteranno eseguibili tutti i comandi che si saranno preventivamente copiati nella directory **C** di sistema, in modo autonomo o sfruttando il programma **ArpInstall**, che provvede ad installare automaticamente tutti i comandi e le risorse di Arp.

Problemi di rad

Sono passato da un Amiga 500 ad un 2000 con 1 Megabyte di chip ram, e non riesco più ad installare una Ramdrive recuperabile (Rad) resistente al reset. Ovvero, la Rad si installa, ma dopo il reset il sistema ne ignora la presenza. Se possibile, vorrei anche sapere il significato dei simboli posti frontalmente nel tastierino numerico.
(Giampiero - Roma)

Due risposte molto brevi. Per quanto riguarda la Rad, il problema è proprio legato alla configurazione di memoria con **un Megabyte tutto di Chip Ram**. In questa condizione, oltre alle normali procedure di installazione, è indispensabile che venga prima eseguita una istruzione **Setpatch R**, preferibilmente da inserire nella startup sequence. Si badi che, nel file di avviamento del disco Workbench 1.3, un comando Setpatch è normalmente presente, ma non sempre seguito dal parametro **R**.

Quanto alla seconda domanda, cui peraltro si è già risposto in altra occasione, si tratta semplicemente di simboli utilizzabili in emulazione Ms-dos, nel caso di (vecchi) programmi che non prevedano l'uso di una tastiera estesa, ovvero dotata di specifici tasti per il movimento del cursore, per la paginazione, eccetera.

Shell e icone

Non c'è un modo, da Shell, per creare una directory col tipico disegno a cassettino?
(A. Giannone - Roma)

La domanda è un po' ambigua, ma vediamo di venirne a capo ugualmente.

Da ambiente Shell, l'unico modo per creare una directory (escludendo casi particolari) consiste nell'adoperare il comando **Makedir**. In tal modo, però, l'unico modo per visualizzarla consiste nell'impartire **List** (oppure **Dir**) sempre da ambiente Shell, mentre da Workbench non sarà possibile accedervi, né tantomeno vederne la classica icona a cassetto, che **non** viene creata assieme alla directory.

```
;crea una directory da Shell con
;associata una icona-cassetto
.Key nome
If <nome>X eq X
 Echo "Manca il nome della directory!"
 Quit
Endif
Makedir <nome>
Copy Workbench1.3:empty.info <nome>.info
```

L'icona, d'altra parte, non è che un file di nome analogo a quello della directory creata, ma con suffisso "**.info**", e caratteristiche particolari che ne fanno una icona di tipo "**Drawer**". Collegare questo tipo di icona ad una directory, non costituisce certo un problema: basta prelevarne una qualsiasi (non necessariamente con disegno a cassetto) purché di tipo Drawer, e copiarla con il nuovo nome.

Per chi è alle prime armi, e non vuole cimentarsi in imprese complicate (mica tanto, però...), una icona del genere è sempre a disposizione: la directory **Empty** del disco Workbench. Si supponga di voler creare una directory di nome **Test** nella **Ram Disk**. Basterà in pratica impartire un comando...

```
Makedir Ram:test
```

...e farlo poi seguire da...

```
Copy Workbench1.3:
Empty.info Ram:Test.info
```

...(digitato su un unico rigo) per avere tanto la directory che la relativa icona-cassetto manipolabile anche a suon di mouse. Piuttosto che ripetere questa procedura ogni volta, è consigliabile farsi un comando apposito, ovvero editare un brevisimo file batch di nome (per esempio) **Mkdir**, strutturato come appare nel riquadro a parte.

Una volta copiatolo con un qualsiasi editor **Ascii** (in mancanza d'altro, con **Ed**) e memorizzatolo nella directory **S**, può essere eseguito con...

```
Execute Mkdir Nomedir
```

...con **Nomedir** eventualmente comprensivo di percorso. In ogni caso, oltre alla directory, verrà creata anche una icona-cassetto identica a quella visibile con nome Empty nel disco Workbench. Se anche quest'ultimo non fosse presente nei drive, un requester di sistema ne chiederà l'inserimento al momento opportuno.

Come ovvio, nulla vieta che il file-icona venga prelevato da qualche altro disco, inserendone il nome al posto di quello da noi adoperato nel batch file (Workbench1.3). Non si dimentichi inoltre che, adoperando sul file batch un comando...

```
Protect Mkdir +S
```

...lo si renderà autoeseguibile, per cui potrà essere trasferito nella directory C, e invocato esattamente come se si trattasse del normale Makedir, vale a dire senza adoperare Execute.

Chi fosse proprio alle prime armi, si limiti a mettere in pratica quanto appena detto, magari andando a ripescare (se il caso dal nostro servizio arretrati) il numero **77** della rivista, che spiega la strutturazione di un file batch.

Richieste prezzi

Ho letto, sulla vostra rivista, la recensione su xxxx; potreste dirmi il prezzo del prodotto?

Visto l'incremento di lettere del genere, che vanno ad

Un tasto... ricorrente

Come si ottiene il carattere di esponente (^) che non riesco a trovare sul mio Amiga con tastiera italiana?
(Giacomo Longo - Messina)

Come già scritto in altre occasioni, è necessario mantenere pressati contemporaneamente i tasti **Shift** ed **Alt**, ed agire sul tasto **6** posto in cima alla tastiera (**non** sul tastierino numerico).

Tanta, troppa roba

Gradirei dei chiarimenti sulle Librerie, sulla Startup-Sequence e sulle Directory: a cosa servono? Siccome presumo che tali argomenti siano già stati trattati, mi potreste fornire quali numeri arretrati acquistare? Perché non pubblicate una paginetta con elencati i numeri arretrati e gli argomenti in essi trattati?
(E. Manzoni - Pioltello)

La risposta ad una simile domanda, come intuito dallo stesso lettore, richiederebbe almeno un paio d'anni di pubblicazioni mensili. Il che, però, non vuol dire che siano argomenti particolarmenti complessi: se si esclude l'uso delle librerie di sistema, che certamente richiedono un impegno non proprio irrisorio, e che comunque vanno affrontate con già una certa dimestichezza alle spalle, gli altri temi accennati rientrano in fin dei conti nel vasto ambito di Amigados, ampiamente trattato (anche in dettaglio) sulla nostra rivista.

Il consiglio è immediato: consultare quanti più articoli possibile che riguardano il mondo "facile" di Amiga, ovvero il **Dos** e il **Basic**, e solo in un secondo tempo prendere contatto con il più complesso ambiente fatto di librerie e linguaggi non proprio semplici come il C o l'Assembler. Tutto ciò, sia ben chiaro, se non si ha alle spalle alcuna esperienza computereccia. In caso contrario, potrebbe anche essere possibile un "salto" immediato verso lidi più ostici.

Impossibile riassumere qui i numeri della rivista che si sono occupati di Dos, Basic e Librerie: praticamente quasi tutti, da quando si è cominciato a parlare di Amiga. In particolare, dal n. **75** al n. **82** è stato sviscerato tutto il possibile su Amigados, ma anche altri articoli, apparentemente rivolti ad applicazioni differenti, possono contenere informazioni utili per giungere ad una comprensione globale del computer.

La proposta di pubblicare una specie di sommario generale potrebbe essere una idea fattibile, si vedrà, comunque qualcosa del genere è già presente nei floppy **Amigazzetta** che saltuariamente vengono pubblicati dalla nostra Editrice.

Cominciare non è facile... ma neanche impossibile.

Solo da copiare

Vorrei sapere come realizzare il gioco di pag.68 della rivista n.81. Forse inserendo il dischetto Extra?
(Damiano Spinosa - Gaeta)

Il listato cui fa riferimento la lettera ("Indovina, indovinello"), contiene in pratica solo 3 righe non compatibili con Amigabasic, riservate a chi adopera Quick Basic su computer Ibm compatibili. Escludendo queste tre righe, peraltro super evidenziate il testa al listato, il resto va semplicemente copiato nell'editor di Amigabasic.

Ovvero, vista la perplessità del nostro lettore alle prime armi, occorre:

* Attivare Amiga col normale disco Workbench.
* Inserire il disco Extras, aprirne l'icona, e biclickare su Amigabasic.
* Copiare il listato nella finestra di editing (list).

A queste banali istruzioni, ne va però aggiunta una fondamentale: prima di ogni altra cosa, dare almeno un'occhiata superficiale alla manualistica fornita in dotazione al computer. Non sarà il massimo della chiarezza, ma le informazioni più elementari in fondo sono sufficientemente trattate.

intasare i cestini strategicamente disposti in redazione, è d'obbligo un consiglio:

telefonare direttamente alle ditte che commercializzano il prodotto, chiaramente indicate con tanto di indirizzo e numero telefonico nelle pagine della rivista.

Non si tratta di cattiva volontà (...), ma di fatto i prezzi cambiano spesso (di solito, **in ribasso**) anche nel giro di una settimana, per non parlare di eventuali sconti od occasioni.

Su questi ultimi, pertanto, solo un contatto diretto può garantire la tempestività d'informazione.

Dos e Append

Come posso fare ad "incollare" altro testo in fondo ad un file già esistente senza perderne il contenuto?
(Marcello Fiaschi - ??)

Il tema, a quanto pare, è piuttosto ricorrente, nonostante ne sia stato fatto cenno nella rubrica Amigafacile, trattando del comando **Join**. Che, comunque, non è l'unico a garantire il tipo di prestazione richiesta dal lettore. Vediamo di chiarire l'argomento, visto che le situazioni nelle quali ci si può imbattere sono più d'una.

La più banale è quella in cui sono già presenti due file di testo, che chiameremo **File1** e **File2**. In questo caso, la soluzione più ovvia è impartire...

```
Join file1 file2 To File3
```

...ed il **file3** sarà la fusione dei due precedenti. Si può anche desiderare, in modo diretto oppure da batch file, aggiungere una singola riga, o più righe, per esempio al File1. Stavolta, Join da solo non basta.

Una soluzione potrebbe essere quella di creare un file provvisorio redirigendo normalmente l'output di un comando come **Echo** verso un file fittizio, da appendere in un secondo tempo tramite Join. Per essere più chiari, si proceda come segue.

Prima di tutto, ci si porti in **Ram Disk** per operare in questo device virtuale impartendo `Cd Ram:`. Si crei il File1 (che finirà in Ram disk) digitando nella finestra Shell `Copy * File1`. Quando il cursore si sarà bloccato, si digiti una frase qualunque, per esempio "**Io sono il file 1**", e si pressino poi (dopo un Return) in contemporanea i tasti **Ctrl** e barra inversa ("\"). Si avrà così in Ram disk un file-testo di nome File1 (tecnica di redirezione della consolle attuale, più volte affrontata in queste pagine). Naturalmente è solo un esempio, per cui il file di testo potrebbe già essere presente, o comunque essere

creato con tecniche differenti (un ripasso non guasta mai...).

Supponiamo ora di voler aggiungere al File1 un'altra riga. Applicando quanto detto prima, si potrebbe seguire una procedura di questo tipo:

* impartire un comando `Echo >temp "e chi se ne frega?"`. In tal modo si creerà un file di nome **"temp"**, contenente il testo appena digitato.

* impartire `Join file1 temp to file2`. Il file2, come verificabile adoperando un banale `Type File2`, conterrà entrambe le frasi di un ipotetico dialogo... dai toni non proprio gentili.

Adoperando questa tecnica, si badi, il File1 resterà comunque integro ed immodificato.

Esiste poi una terza soluzione, poco documentata nella manualistica di Amigados, che consente un vero e proprio append al file originario (nel nostro caso File1), che risulterà quindi modificato definitivamente (occhio quindi ad eventuali errori). Si tatta di adoperare ancora la redirezione dell'output di Echo, ma inserendo **DUE** volte il simbolo maggiore (**>**). In pratica: ci si assicuri del contenuto del File1 impartendo Type File1. Se tutto è in regola, questo dovrebbe ancora contenere solo la frase "Io sono il file1".

Ora si impartisca da Shell...

```
Echo >>File1 "molto lieto"
```

Tutto fatto. Un ulteriore controllo con Type File1, mostrerà come il file originario abbia acquisito quest'ultimo input senza perdere il precedente contenuto, come ogni Append che si rispetti.

L'uso di una tecnica piuttosto che l'altra dipenderà, come ovvio, dalle esigenze del momento, ma quest'ultima in particolare sarà senz'altro quella che con più frequenza risulterà utile.

Probabile virus

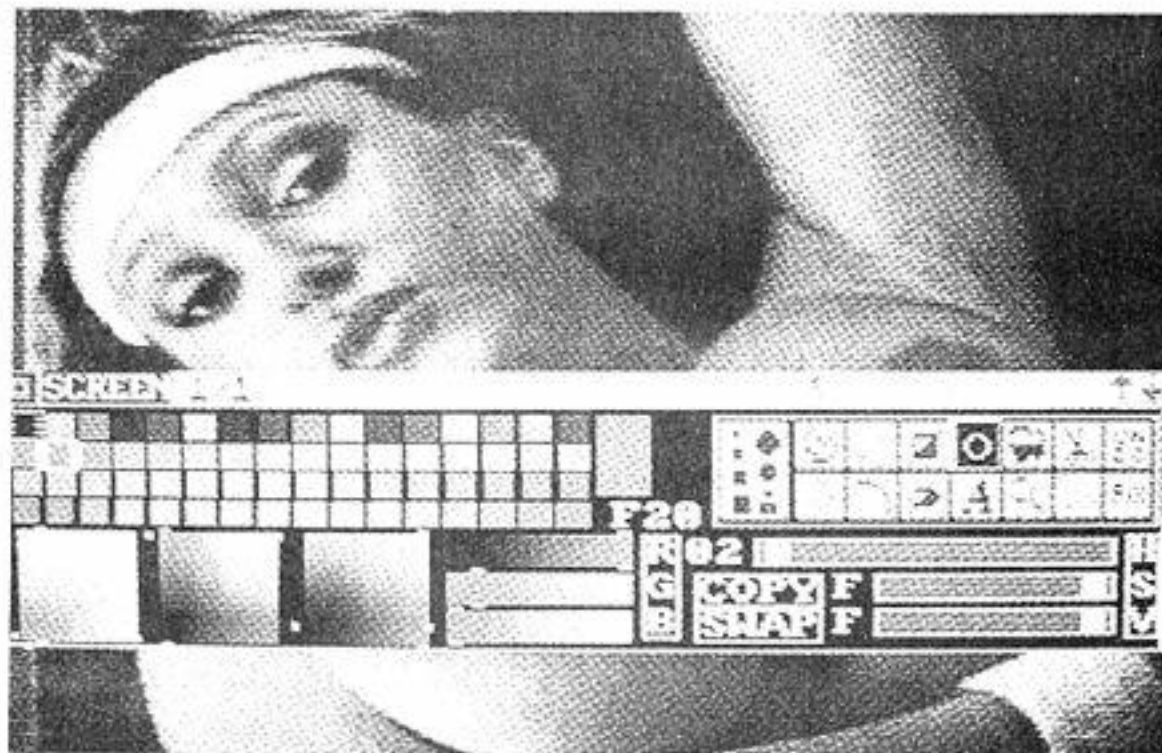
Posseggo un Amiga500 con un solo drive, ed ho notato uno strano comportamento: dopo avere installato un disco adoperando il punto interrogativo finale a causa del drive singolo, questo diventa normalmente bootabile. Il problema sorge invece se voglio togliere l'autoboot. Allo scopo adopero una sintassi **`Install Drive Df0: Noboot ?`** ***(e 2 Return). Ma se subito dopo impartisco*** **`Install Drive Df0: Check ?`** ***(e due Return) il sistema mi segnala che "Appears to be a normal V1.2/V1.3 bootblock"! Ed infatti il disco è rimasto bootabile, come provato dopo aver spento e riacceso Amiga.***
Capita anche ad altri, oppure ho preso un nuovo virus?
(S. Manferdini - S. Lazzaro)

Se le procedure indicate sono state eseguite esattamente come descritte, in effetti il bootblock del disco dovrebbe essere rimosso. Anche se può sembrare superfluo: ci si è ricordati di cambiare floppy prima dei due Return? Dato per scontato che lo si sia fatto, si può provare ad adoperare una tecnica diversa, ovvero copiare il comando Install in Ram Disk (comando `Copy c:install ram:`), inserire il floppy da modificare nel drive, quindi digitare da Shell...

```
Ram:install Df0: Noboot
```

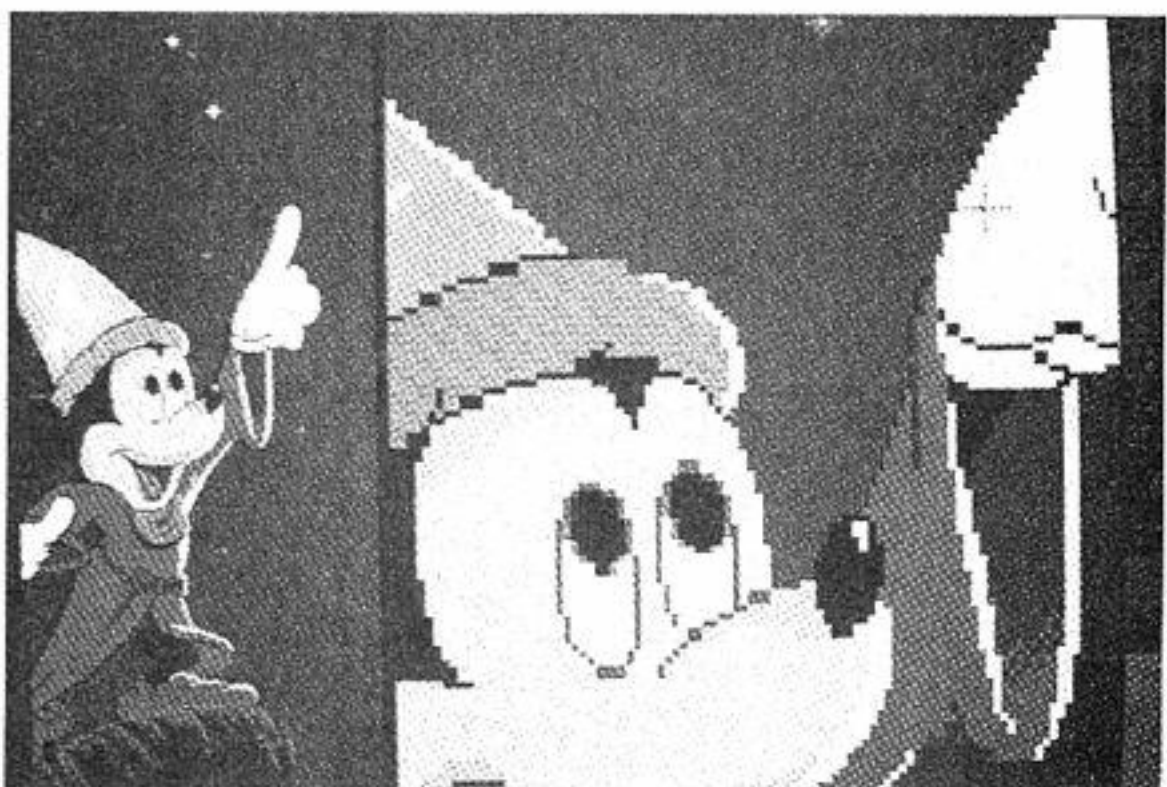
...senza punto interrogativo e seguito quindi da un solo Return.

Questo non perché la tecnica adoperata dal lettore sia scorretta, quanto per verificare che non si sia commesso qualche errore adottandola.

Se anche così il floppy dovesse rimanere bootabile, l'ipotesi più probabile è proprio l'interferenza di un qualche virus, non necessariamente nuovo. Virus che, a meno non sia installato nel disco adoperato normalmente per il boot (il Workbench, di solito), dovrebbe per forza di cose provenire da quello che si intende rendere non bootabile. Come controprova al tutto, si ritenti la procedura adoperando una copia del Workbench sicuramente non "infetta", protetta in scrittura, e un floppy ancora nuovo, mai adoperato. Prima di formattarlo, è però opportuno lanciare un antivirus come **VirusX** (ultima versione al momento in cui si scrive: **4.01**), che continua a restare attivo qualunque operazione si compia, e senza interferire.

Se un virus dovesse tentare l'attacco, verrebbe così smascherato ed eliminato da Virusx, che reinstalla i floppy "infetti" dopo aver segnalato il problema, e richiesto debita autorizzazione. Come ovvio, dopo l'intervento di Virusx, andrebbe comunque riapplicato Install con l'opzione Noboot.

Basic e stampante

Ho un problema che mi assilla: vorrei inviare da Basic l'output verso la stampante adoperando Llist e Lprint, ma vengo sempre bloccato da un criptico messaggio 'File Already Open'.
(Marco Avonto - Novara)

Il messaggio, in effetti, tanto criptico non è. In mancanza di maggiori dettagli, la causa più probabile dell'inconveniente è legata alla mancata chiusura di un file precedentemente aperto verso la periferica **Par:**, **Prt:**, oppure **Lpt1:**.

Si considerino come esempio queste righe:

```
OPEN "prt:" FOR OUTPUT AS 1
PRINT #1, "Ciao!!!"
LPRINT "Come va?"
```

Se mandate in esecuzione, provocheranno appunto l'errore **File Already Open**. Per evitarlo, sarà sufficiente inserire una istruzione `Close 1` immediatamente prima del comando Lprint, e in tal modo verrà stampata tanto la stringa "ciao!!!" che il "come va?"

Le stesse considerazioni, come ovvio, valgono anche se **Llist** oppure **Lprint** sono impartiti in modo diretto dalla finestra di output, e in precedenza si sia fatto "girare" un programma che non provvede a chiudere eventuali files aperti verso la stampante (o la porta parallela).

Difficile hacking

Ho notato che, adoperando Action Replay, alcuni games non possono essere "congelati". Inoltre qualche volta non riesco a modificare il numero di vite di un gioco: attivo il Trainer, cambio il valore delle vite in un certo indirizzo esadecimale, ma queste rimangono le stesse. Riesaminando le locazioni da me cambiate, le ritrovo esattamente come prima del mio intervento...
(Michele Verzi - Firenze)

Il fatto che alcuni games non possano essere "freezati", come pure (almeno in parte) il motivo del ripristinarsi di alcune locazioni di memoria, può essere fatto risalire con ogni probabilità... al programmatore. Detto in altre parole: se chi ha programmato il game ha voluto evitare il più possibile manipolazioni dall'esterno, ci è riuscito.

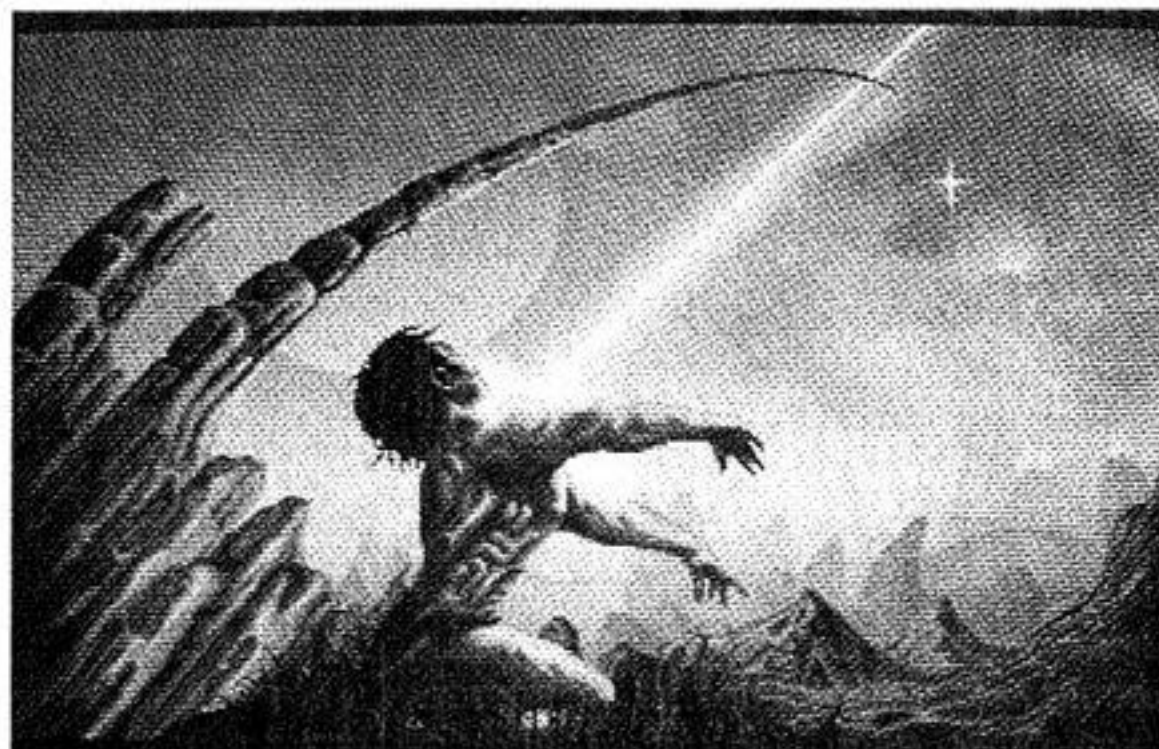
Come del resto accennato nello stesso manuale di Action Replay, non tutti gli interventi sono così immediati come sembrano. Esempio tipico, proprio il tentare di modificare il numero di vite del gioco. Intanto occorre essere certi che siano proprio quelle modificate le locazioni che memorizzano questo dato. A parte alcuni trucchi classici per mimetizzare la realtà (riportati nel manuale di Action Replay), potrebbe proprio darsi che il programma (il game) reinizializzi quelle locazioni dopo lo start.

In questo caso si renderebbe necessario andare a trovare quella sezione di programma, e modificarla. Cosa di certo non facile, ma non si può pensare di ritrovársi certe cose servite su un piatto d'argento...

Baco di Amos?

Ho acquistato il Basic Amos con entusiasmo per programmare il mio (vecchio) Amiga 1000, ma sono stato bloccato da un problema che non sono riuscito a risolvere. Usando le variabili reali (che in Amos non sono quelle di default, ma vanno identificate dal simbolo "#" che segue il nome della variabile) se nel programma è presente un input qualsiasi dopo l'uso di queste variabili, il computer si blocca e bisogna resettare tutto!
(A. Nardozza - S. Aurunca)

La lettera prosegue proponendo un listatino di esempio così strutturato...

```
Screen Open
1,640,256,8,Hires
10 Cls
Input "Dammi 3 numeri
";A#,B#,C#
Print A#*B#*C#
Input "Ancora (s/n)? ";R$
If R$="s" Then 10
Screen Close 1
```

...che funziona perfettamente se al posto delle variabili reali si adoperano le variabili intere, mentre in questa forma il computer si blocca alla richiesta se continuare o meno.

Il problema, abbiamo riscontrato, esiste realmente e non è legato all'uso di Amiga 1000, ma si verifica anche su modelli più recenti.

Come giustamente fa notare il nostro lettore nella lettera, un limite di questo tipo non impedisce di trattare elementi come la grafica, gli schermi, eccetera, ma certo blocca iniziative didattiche volte, per esempio a risolvere equazioni, o comunque applicazioni che richiedano una certa precisione di calcolo: Amos è sì un Basic volto soprattutto alla realizzazione di games, ma dovrebbe essere possibile accedere anche ad altre applicazioni, magari supportate da una interfaccia grafica dalle mirabolanti prestazioni, considerate le potenzialità di questo linguaggio.

Va precisato, comunque, che l'inconveniente (vogliamo chiamarlo Bug?) si presenta solo nella versione **1.2** di Amos, mentre con la precedente **1.1** lo stesso sembra non sussistere.

Ci affrettiamo ad aggiungere "sembra" in attesa di vagliare più a fondo questo interprete, che sembra riscuotere un notevole successo.

Se nel frattempo qualche altro lettore ha trovato una soluzione alternativa alla cosa, non ha che da farcelo sapere: saremo ben lieti di ricrederci (e pubblicarne il contributo).

Basic e Dos

Come posso far eseguire un comando della directory C (come Assign, List, eccetera) dal Basic?
(Armando Rampelli - Roma)

Dipende dal Basic adoperato. Nel senso che alcuni interpreti sono dotati di una propria gestione del Dos, come per esempio il **Blitz Basic** (in realtà da considerarsi un Basic compilato) che addirittura consente di accedere al **Cli** senza uscire dal proprio ambiente. Oppure come **Amos**, che, pur non disponendo di una simile feature, tuttavia implementa una vasta gamma di comandi complementari a quelli del dos.

Se invece si ha a che fare con il Basic Microsoft, (ahinoi) meglio conosciuto come **AmigaBasic**, o si ha comunque l'esigenza di mandare in esecuzione un comando esterno, non c'è altra soluzione che ricorrere alla funzione **Execute** della Dos Library: semplice da usarsi, ma che richiede alcune attenzioni particolari.

Intanto, cominciamo col vederne la sintassi, riferendoci ad un suo uso in ambito Amigabasic:

```
ok%=Execute%(c&,in&,out&)
```

Trattandosi di una funzione, come si vede non va richiamata con **Call**, ma associata ad una variabile (nel nostro caso **ok%**). I parametri sono da interpretarsi così:

* **C&** - Puntatore all'indirizzo di memoria ove è memorizzata la stringa che identifica il comando. In Amigabasic, questo valore lo si otterrà adoperando l'istruzione **Sadd**. La stringa, come di norma quando si tratta con Amiga ad un livello più basso, deve obbligatoriamente concludersi con un **Chr$(0)**.

* **In&** - Questo valore rappresenta l'input, o meglio il suo handle, che poco riguarda il basic. In pratica, andrà posto sempre a zero, in quanto l'input è precisato dalla stringa-comando.

* **Out&** - Con questo parametro si specifica l'output verso il quale dirigere il comando. Da un punto di vista pratico, nella maggior parte dei casi sarà rappresentato da un puntatore ad una stringa (anche questa con un Chr$(0) finale) che identifica il nome di un file, oppure il device **Nil:** del Dos. Volendo, si potrebbe porre a zero il valore di questa variabile, nel qual caso l'output verrebbe diretto verso la cosiddetta console corrente, ma attenzione: questa possibilità richiede obbligatoriamente che Amigabasic sia stato lanciato da Shell (o Cli), e non da Workbench. In tal caso, se per esempio si assegnasse **List** alla stringa-comando, e si assumesse zero come puntatore all'output, il listing della directory corrente verrebbe mostrato nella finestra Shell dalla quale si è lanciato Amigabasic. Se, invece, si fosse adoperato il solito doppio click da Workbench per attivare l'interprete, azzerando questo parametro si otterrebbe solo una visita dell'amico **Guru**, in quanto non esisterebbe una console del Dos per l'output (la finestra del basic non lo è!).

Dopo tanta teoria, si dia un'occhiata al breve listato pubblicato a parte: si noterà come tutto è più facile di quello che sembra.

Occorrerà naturalmente provvedere alla dichiarazione delle funzioni adoperate (**Execute** e **Open**), e far sì che il Basic possa accedere al file **Dos.bmap**, per chi ancora non lo sapesse rintracciabile nella directory Basicdemos del disco Extras. Nel listato di esempio, viene proposta una subroutine che può essere inserita in qualsiasi altro listato: basterà passargli una stringa col nome del comando ed una con il nome del file o device ove dirigere l'output, ed al resto penserà lei, ivi compresa l'aggiunta del Chr$(0) alle stringhe.

```
REM esempio di uso del Dos da Basic
'-----------------------------------
DECLARE FUNCTION xOpen& LIBRARY
DECLARE FUNCTION execute% LIBRARY
LIBRARY "dos.library"
'-----------------------------------
comando$="newshell"
redir$="nil:"
CALL dos (comando$, redir$)
LIBRARY CLOSE
'-----------------------------------
SUB dos (istr$, outp$) STATIC
 istr$=istr$+CHR$(0)
 outp$=outp$+CHR$(0)
 punt&=xOpen&(SADD(outp$),1006)
 exe%=execute%(SADD(istr$),0,punt&)
 CALL xclose(punt&)
END SUB
```

Se mandato in esecuzione così com'è, il programmino si limita ad aprire una nuova finestra Shell, ed a concludere il suo operato. Per questo motivo, l'output viene diretto verso Nil: (il nulla di niente, per intenderci), visto che il comando **Newshell** non necessita di un output su schermo. Se, invece, avessimo voluto adoperare (per esempio) **List**, saremmo stati costretti ad assegnare il nome di un file alla variabile **Redir$**, eventualmente comprensivo di percorso.

Si provi a modificare le assegnazioni del programma in modo da avere...

```
comando$ = "List"
redir$ = "Ram:myfile"
```

Stavolta,dopo l'esecuzione, sarà presente in Ram Disk un file di nome Myfile, contenente l'elenco dei file della directory corrente, esattamente nello stesso formato che si avrebbe impartendo List da una finestra Dos. Come ovvio, per eventuali usi da Basic, questo file andrebbe poi trattato come più aggrada dal programma, ovvero a suon di Open, Print#, Input#, eccetera.

Nella subroutine proposta, in funzione soprattutto della più larga compatibilità possibile, il file di output viene aperto (nonché richiuso) sfruttando le funzioni della Dos Library **xOpen** e **xClose**, il cui uso (molto semplice) può essere dedotto direttamente dal listato. Come ovvio, il tutto presuppone che il comando invocato da Basic sia presente nel percorso di ricerca del dos (tipicamente la directory **C**), o nella directory corrente, mentre è sempre obbligatoria la presenza di **Run** nella directory C. Nulla vieta, comunque, che nella stringa che indica il comando si precisi ove è rintracciabile il comando (per esempio: Sys:system/format), nel qual caso non è più indispensabile che questo si trovi compreso nel Path di ricerca.

Per il resto, non c'è che da dilettarsi in tentativi anche azzardati: è l'unico modo per trovare applicazioni degne di nota a quanto finora espresso.

In bocca al Guru...

di Fabrizio Bazzo

Cifriamo un testo ASCII ricorrendo alla matematica

Vi siete mai chiesti a che servono le matrici che, a scuola, fanno spesso venire il mal di testa? Ecco una pratica applicazione

Ognuno di noi, si sa, ha qualcosa da nascondere; quando non ci si può affidare solo alla memoria e questo qualcosa (messaggi spionistici, progetti rivoluzionari o, più realisticamente, il codice del **Bancomat**) ha bisogno di essere messo per iscritto, il pericolo che diventi di dominio pubblico spinge a cercare soluzioni strane.

Leonardo da Vinci, per esempio, scriveva i suoi progetti da destra verso sinistra, per rileggerli in seguito mediante uno specchio; oggi esistono metodi più comodi, tra cui la codifica mediante computer.

Metodi di codifica ve ne sono in abbondanza, alcuni sicuri, altri un po' meno; forse qualcuno, non più giovanissimo, si ricorda dell'orologio codificatore delle **Giovani Marmotte** che si limitava a stabilire una corrispondenza biunivoca tra un alfabeto ordinato (A-Z) e uno "rimescolato"; ad ogni A corrispondeva, ad esempio, una Y, e viceversa; il grave difetto di un simile codice è che se nel messaggio originale compaiono cinque **A**, ci saranno altrettante **Y** in quello cifrato, per giunta nella stessa posizione; risalire al messaggio originale, pertanto può risultare fin troppo semplice.

Un altro metodo, molto sfruttato ancora oggi, consiste nel "gonfiare" il messaggio originale con parole o lettere senza senso, riservandosi di rileggerlo con cadenze prefissate o, meglio, attraverso un foglio di cartone forato nei punti giusti; anche in questo modo, però, il messaggio originale resta esposto, anche se non immediatamente visibile.

La sostituzione dell'alfabeto tradizionale con il codice ASCII e l'abbinamento dei due metodi appena descritti può ingarbugliare ancora di più le cose, ma un curioso molto testardo potrebbe sempre venirne a capo, e noi non vogliamo correre rischi.

Altro difetto, per nulla trascurabile in codici di questo tipo, è che una volta nota la chiave di codifica è automaticamente svelata anche quella di decodifica.

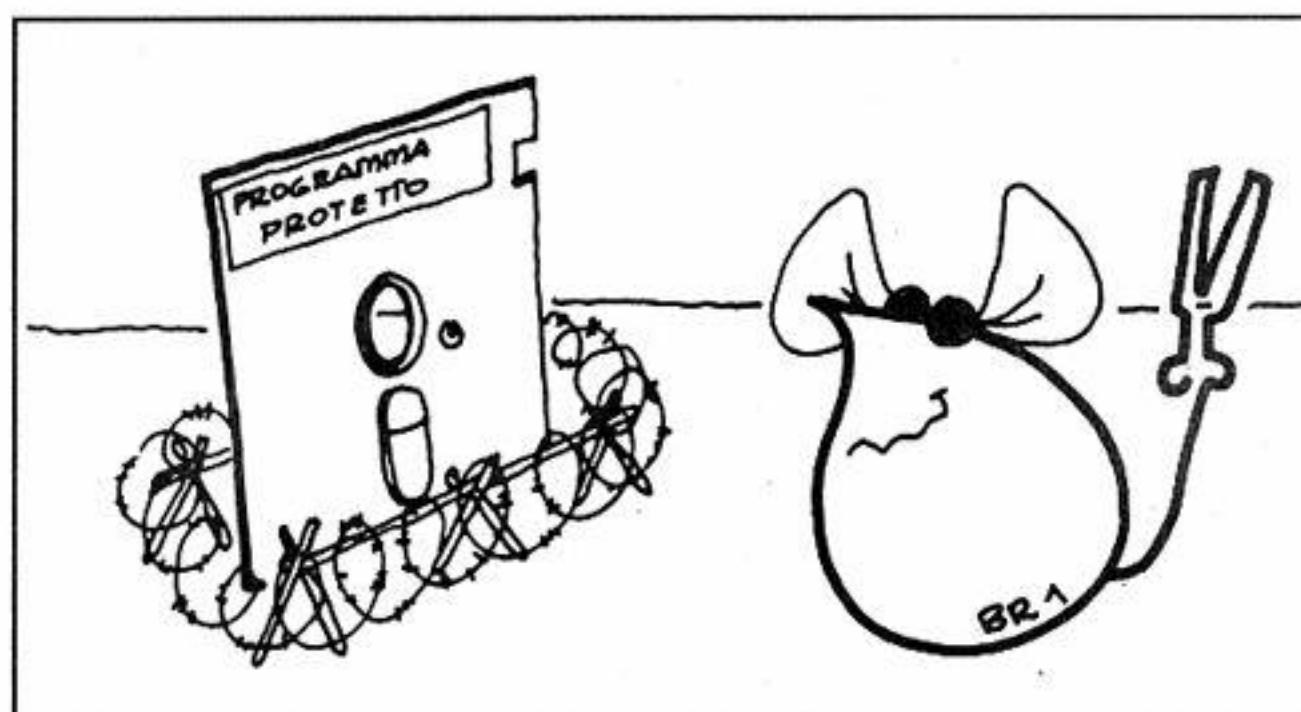
La teoria

I **codici a matrice**, su cui si basa il programma proposto, sono la naturale evoluzione dei **codici a sostituzione**, molto più efficienti di questi: il numero dei caratteri del messaggio da cifrare non viene aumentato artificiosamente e la frequenza relativa di ogni carattere nel

```
      5    *     1/5     =    1

    3 1          2 -1         1 0
    5 2    *    -5  3    =    0 1

 1  0 -1     1  2 -1          1 0 0
 2  1  3  * -2 -9  5     =    0 1 0
 4  2  5     0  2 -1          0 0 1
```

Tabella 1
Relazione tra una matrice e la corrispondente inversa.

messaggio originario è completamente stravolta nel messaggio cifrato.

La lettera **A** potrebbe, infatti, essere tradotta la prima volta con una parentesi **graffa** aperta, la seconda con **q**, e via ingarbugliando. Inoltre il passaggio dalla chiave di codifica a quella di decodifica non è per niente immediato.

Prima di proseguire abbiamo bisogno di scomodare la matematica, e in particolare l'**algebra lineare**: niente paura, solo lo stretto indispensabile per capire di che cosa stiamo parlando, e senza scomodare teoremi che, sebbene utilissimi, non abbiamo il tempo (nè la voglia) di spiegare.

Tabella 2 (codifica e decodifica)
Il messaggio è scomposto nei corrispondenti caratteri ASCII, che sono diminuiti di 31 per evitare di pescare in seguito valori ASCII inferiori a 32. Viene quindi eseguito il prodotto riga per colonna con la matrice di codifica. Il range dei caratteri utilizzato e' da 32 a 126 (95 caratteri). Volendo, e' possibile limitarsi, ad esempio, alle sole lettere maiuscole (65 - 90 => MOD 26).

```
1 3   C o m m o d o r e   C o
2 5   m p u t e r   C l u b !

1 3    67 111 109 109 111 110 111 114 101  32  67 111
2 5   109 112 117 116 101 114  32  67 108 117  98  33   ASCII...

1 3   36 80 78 78 80 69 80 83 70  1 36 80
2 5   78 81 86 85 70 83  1 36 77 86 67  2                ... - 31...

1*36+3*78  1*80+3*81  1*78+3*86  1*78+3*85
2*36+5*78  2*80+5*81  2*78+5*86  2*78+5*85 .....     ..prodotto
                                                        riga
270 323 336 333 290 318  83 191 301 259 237  86         per
462 565 586 581 510 553 165 346 525 432 407 170       colonna...

 80  38  51  48   5  33  83   1  16  69  47  86
 82  90  16  11  35  78  70  61  50  52  27  75      ...mod 95...

111  69  82  79  36  64 114  32  47 100  78 117
113 121  47  42  66 109 101  92  81  83  58 106      ...  + 31...

o E R O $ @ r   / d N u
q y / * B m e \ Q S : j                              ... ASCII
```

Posto che la chiave di codifica da adottare sarà una **matrice quadrata** di interi, questa sarà una chiave valida solo se il suo **determinante** sarà diverso da zero. Tale particolare garantirà che la matrice è invertibile; ma ancora non basta. Occorre che l'inversa sia ancora una matrice di interi (cfr. tabella 3).

Quando una matrice quadrata ha il determinante diverso da zero se ne può determinare l'inversa eseguendo una serie di cosiddette operazioni di riga, fino a trasformarla nella **matrice identica** (che, detto in soldoni, è l'**uno** del mondo matriciale); eseguendo le stesse operazioni su una matrice identica si ottiene l'inversa della matrice di partenza (vedi tabella 1). Occorre poi definire il **prodotto tra matrici**, dato che il messaggio da codificare è esso stesso una matrice di codici ASCII.

Il prodotto è possibile solo se il numero di colonne della prima matrice (in questo caso la "chiave") è uguale al numero di righe della seconda matrice (cfr. tabella 2).

Ricapitolando

Per crittografare un messaggio si prende in considerazione una matrice quadrata di interi che abbia, come inversa, ancora una matrice di interi.

Si dispongono, poi, i caratteri del messaggio in modo da formare una matrice che abbia un numero di righe uguale al numero di colonne della matrice chiave,

```
3 1
5 2  =  3 * 2 - 1 * 5      definizione

1 4 7
2 5 8          5 8          2 8          2 5
3 6 9  =  +1 * 6 9   -4 *   3 9   +7 *   3 6
```

Tabella 3 (determinante)
Per matrici quadrate di ordine superiore a 2 il procedimento è ricorsivo e presenta alteranza di segni a partire dal più (+) e muovendosi verso il basso oppure verso destra. Nel programma non è stata seguita questa tecnica, ma un trucco consentito da alcuni teoremi: se al di sotto della diagonale vi sono solo valori nulli il determinante è dato dal prodotto degli elementi presenti sulla diagonale stessa.

```
'codici  a matrice
'by Fabrizio Bazzo

'menu enable
DATA "Matrix", "New   ", "Deter.", "Invert", "Load   ", "Save   "
DATA "Code  ", "Text  ", "DeCode", "Load   ", "Save   ", "Quit   "
FOR i = 1 TO 2: FOR j = 0 TO 5: READ m$
MENU i, j, 1, m$: NEXT: MENU i + 2, 0, 0, "": NEXT
ON MENU GOSUB choise: MENU ON: sp$ = SPACE$ (55)
DIM SHARED i (20, 20)
DIM m (20, 20), ml (20, 20), d (20, 50), c (20, 50)

main:
WHILE vogliadistudiare = 0      'loop infinito...
  IF refresh THEN
    refresh = 0: l = LEN (msg$) : CLS: w = 0
    IF k OR l THEN     'c'e' qualcosa da stampare?
 IF k = 0 THEN k = INT ( (l^2) ^ (1/5) )ELSE w = 1 'split mess.
      GOSUB where: FOR z = y TO y + k-1: LOCATE z, x
      IF w THEN
        FOR i = 1 TO k: PRINT USING"##.#"; m (CSRLIN-y + 1, i) ;
         NEXT: PRINT SPC (3) ;
      END IF
      IF l THEN PRINT MID$ (msg$, ( (CSRLIN-y)  * q + 1), q)
      NEXT: GOSUB depth
    END IF
  END IF
WEND

where:      'posizionamento
y = INT ( (24-k) /2) : q = INT (l/k + .9)
x =  (76-q-w *  (3 + k * 4) ) /2: RETURN

depth:      'bassorilievo
x1 =  (x-2)  * 8: y1 =  (y-2)  * 8
dx =  (q + w *  (k * 4 + 3)  + 2)  * 8: dy =  (k + 2)  * 8
FOR c = 0 TO 1
  LINE  (x1, y1 + dy * c) - (x1 + dx, y1 + dy * c), c + 1
  LINE  (x1 + dx * c, y1) - (x1 + dx * c, y1 + dy), c + 1
NEXT: RETURN

choise:     'gestione menu
ON MENU (0)  GOSUB matrix, code: RETURN

matrix:
ON MENU (1)  GOSUB newm, determ, invert, loadm, savem: RETURN

code:
ON MENU (1)  GOSUB text, decode, loadt, savet, quit: RETURN

newm:   'ingresso dati matrice
CLS: f = 1: PRINT : INPUT "Ordine della matrice quadrata "; k
PRINT "Inserire i valori per colonne":PRINT
FOR j = 1 TO k: FOR i = 1 TO k
```

Il listato (parte 1)

e si esegue il prodotto delle due matrici. Con opportuni controlli i valori della matrice prodotto saranno ancora coerenti con il codice ASCII.

Si ottiene così un messaggio cifrato, in grado di procurare al nostro curiosone un robusto mal di testa.

Per riottenere il messaggio originale basta moltiplicare l'inversa della matrice chiave per la matrice del messaggio cifrato.

Il programma

I calcoli di prodotti e inversioni sono lunghi e noiosi, perciò li faremo fare al computer (in fondo serve proprio per questo).

Una volta digitato, salvato e lanciato il programma, si potrà ammirare un bello schermo vuoto, mentre un loop attende la scelta, da parte nostra, di un elemento del menu.

Dal menu **Matrix** (primo in alto a sinistra che appare premendo il tasto destro del mouse) selezioniamo **New** (nuova

```
  PRINT "riga "i" colonna "j; : INPUT m (i, j)
NEXT: NEXT
WHILE f  'controllo errori
  CLS: LOCATE  (24-k) /2, 1: GOSUB showmat
  PRINT TAB (33) ; : INPUT"Correzioni "; r$
  IF r$ = "s" THEN GOSUB correct ELSE f = 0
WEND: refresh = 1
RETURN

showmat:
FOR i = 1 TO k: PRINT TAB ( (76-k) /2) ; : FOR j = 1 TO k
PRINT  USING"##"; m (i, j) ; : NEXT: PRINT: NEXT: PRINT: RETURN

correct:
PRINT TAB (10) "Riga e colonna da modificare  < < i, j  > >  ";
INPUT i, j
PRINT  TAB (26) "Nuvo valore "; :INPUT m (i, j) : RETURN

determ:      'calcolo e visualizzazione determinante
FOR i = 1 TO k: FOR j = 1 TO k: ml (i, j)  = m (i, j)
NEXT: NEXT
CALL det  (ml (), k) : LOCATE 5, 25
PRINT USING "Determinante  =  ###.##"; dtm
BEEP: FOR t = 1 TO 5000: NEXT: LOCATE 5, 25: PRINT sp$
RETURN

invert:       'inversione matrice
CALL inv (m (), k) : refresh = 1
FOR i = 1 TO k: FOR j = 1 TO k: m (i, j)  = i (i, j) : NEXT:
NEXT
RETURN

text:         'messaggio
CLS: a$ = "": msg$ = "": LOCATE 5, 1: PRINT "Messaggio: >  ";
WHILE a$ <> CHR$ (13)
 a$ = INKEY$: y = CSRLIN: x = POS (0)
 IF a$ <> CHR$ (13)  AND a$ <> "" THEN
  IF ASC (a$) >  = 32 OR ASC (a$)  < = 126 THEN msg$ = msg$ + a$
  IF ASC (a$)  = 8 THEN  'controllo backspace
   IF LEN (msg$)  = 0 THEN a$ = ""
   IF POS (0)  = 1 THEN LOCATE y-1, 76
   msg$ = LEFT$ (msg$, (LEN (msg$) -2) )
  END IF
 END IF
 IF x < = 75 THEN PRINT a$;  ELSE LOCATE y + 1, 1: PRINT  a$;
WEND: l = LEN (msg$) : refresh = 1
RETURN

decode:    'codifica/decodifica
l = LEN (msg$) : q = INT (l/k + .9)
FOR i = 1 TO k: FOR j =  1 TO q
 z = j + q *  (i-1) : d (i, j)  = 0
 IF z < = l THEN w = ASC (MID$ (msg$, z) ) : c (i, j)  = w-31
NEXT: NEXT
```

Il listato (parte 2)

matrice) e rispondiamo con **2** alla richiesta dell'ordine (dimensione della matrice); quindi inseriamo diligentemente i valori della matrice come in tabella 2. In altre parole...

Riga 1, colonna 1? **1**
Riga 1, colonna 1? **1**
Riga 2, colonna 1? **2**
Riga 1, colonna 2? **3**
Riga 2, colonna 2? **5**

Dopo eventuali correzioni, la nostra matrice apparirà più o meno al centro dello schermo.

Per verificare una cosa che già sappiamo, selezioniamo **Deter** (sempre dal menu Matrix) per calcolarne e visualizzarne il determinante (valore: **-1**).

Affianchiamo ora il messaggio: dal menu **Code** selezioniamo **Text** e copiamo (su un'unica riga e premendo *Return* alla fine) il fantasioso messaggio in figura 2, o qualsiasi altro; in questa fase è possibile utilizzare tutti i caratteri dal **CHR$(32)** al **CHR$(126)**, utilizzando il Backspace per eventuali correzioni.

```
FOR i = 1 TO k: FOR j = 1 TO q
FOR y = 1 TO k    'prodotto riga  *  colonna
 d (i, j)  = d (i, j)  + m (i, y)  * c (y, j) : NEXT
 d (i, j)  = d (i, j) MOD 95 -95 * (d (i, j) <0) 'ASCII 32/126
NEXT: NEXT
msg$ = "": FOR i = 1 TO k: FOR j = 1 TO q
 msg$ = msg$ + CHR$ (d (i, j)  + 31)     'ricostruzione
NEXT: NEXT: refresh = 1
RETURN

savem:
GOSUB nomefile
OPEN file$ + ".mat" FOR OUTPUT AS #1: WRITE# 1, k
FOR i = 1 TO k: FOR j = 1 TO k: WRITE# 1, m (i, j)
NEXT: NEXT: CLOSE #1
RETURN
loadm:
GOSUB nomefile
OPEN file$ + ".mat" FOR INPUT AS #1: INPUT #1, k
FOR i = 1 TO k: FOR j = 1 TO k: INPUT #1, m (i, j)
NEXT: NEXT: CLOSE #1
RETURN
loadt:
GOSUB nomefile
OPEN file$ + ".txt" FOR INPUT AS #1: INPUT #1, msg$: CLOSE #1
RETURN
savet:
GOSUB nomefile
OPEN file$ + ".txt" FOR OUTPUT AS #1: WRITE#1, msg$: CLOSE #1
RETURN
nomefile:
LOCATE 5, 20: INPUT"Nomefile -suffix will be added- "; file$
LOCATE 5, 20: PRINT sp$: IF file$ = "" THEN RETURN main
refresh = 1: RETURN

quit:
MENU RESET: END

SUB det  (m (2), k)  STATIC      'parametri: matrice, dimensione
SHARED dtm                      'ritorna  : dtm
FOR j = 1 TO k-1
  IF m (j, j)  = 0 THEN              'il primo elemento sulla
    r = 1                            'diagonale deve essere  <> 0
    WHILE m (r, j)  = 0 AND r < = k  'se trova nella colonna un
      r = r + 1                      'valore  <> 0...
    WEND
    IF r = k + 1 THEN dtm = 0: EXIT SUB
    FOR s = 1 TO k                   '...somma le due righe
      m (j, s)  = m (j, s)  + m (r, s)
    NEXT
  END IF
  i = 1
  WHILE j + i < = k                 'partendo dall'elemento sulla
    IF m (j + i, j)  <> 0 THEN      'diagonale somma ogni riga alla
```

Il listato (parte 3)

Dopo il Return, le due matrici sono pronte per essere moltiplicate fra loro: ancòra al menu Code selezioniamo **DeCode** per ammirare i guazzabuglio di caratteri che, ora, rappresenta il nostro messaggio; per tornare alle condizioni originare basta selezionare **Invert** (dal menu Matrix) e quindi ancora DeCode.

In generale, ogni qualvolta che si seleziona DeCode viene effettuato il prodotto tra le due matrici visualizzate: ciò significa che un messaggio può essere crittografato un numero indefinito di volte con la stessa matrice chiave, a patto che il prodotto venga moltiplicato **lo stesso numero di volte** per la matrice inversa, se si vuole riottenere qualcosa di leggibile.

Il programma è finalizzato alla visualizzazione sullo schermo, quindi con messaggi molto lunghi occorrono matrici con dimensioni elevate per tenere il tutto sott'occhio; spezzando il messaggio in segmenti di lunghezza opportuna può bastare una piccola matrice 2 x 2 per codificare Guerra e Pace oppure il breve discorso di un qualsiasi parlamentare (di per sè già abbastanza criptico).

```
   alfa =  -m (j + i, j) /m (j, j)  'sottostante, dopo averla
   FOR s = 1 TO k     'moltiplicata per un alfa tale
    m (j + i, s) = m (j, s)  * alfa + m (j + i, s)   'che dopo
       NEXT         ' la somma il valoresottostante sia zero;
    END IF                  'cosi' per ogni colonna
    i = i + 1
   WEND
 NEXT
 dtm = 1: FOR j = 1 TO k
 dtm = dtm * m (j, j) : NEXT 'prodotto diagonale
 END SUB

 SUB inv  (m (), k)  STATIC    'parametri: matrice, dimensione
 FOR i = 1 TO k: FOR j = 1 TO k:      'ritorna: matrice identica
  i (i, j)  = 0: IF i = j THEN i (i, j)  = 1  'modific. l'inversa
 NEXT: NEXT
  FOR j = 1 TO k
  IF m (j, j)  = 0 THEN      'voglio un 1 sulla diagonale
   r = 1
   WHILE m (r, j)  = 0 AND r < = k: r = r + 1: WEND
   FOR s = 1 TO k
  m (j, s) = m(j, s) + m (r, s): i (j, s) =i (j, s) + i (r, s)
   NEXT
  END IF
  alfa = 1/m (j, j)       'e solo zeri sotto
  FOR s = 1 TO k
   m (j, s)  = m (j, s)  * alfa: i (j, s)  = i (j, s)  * alfa
  NEXT
  i = 1
  WHILE j + i < = k      'dall'alto in basso
   IF m (j + i, j)  <> 0 THEN  'e da sinistra verso destra
    alfa = -m (j + i, j)
    FOR s = 1 TO k
     m (j + i, s)  = m (j + i, s)  + m (j, s)  * alfa
 i (j + i, s)  = i (j + i, s)  + i (j, s)  * alfa
    NEXT
   END IF
   i = i + 1
  WEND
 NEXT
 FOR j = k TO 1 STEP -1
 i = 1
  WHILE j-i>  = 1                'dal basso in alto
   IF m (j-i, j)  <> 0 THEN   'e da destra verso sinistra
    alfa = -m (j-i, j)
    FOR s = 1 TO k
     m (j-i, s)  = m (j-i, s)  + m (j, s)  * alfa
 i (j-i, s)  = i (j-i, s)  + i (j, s)  * alfa
    NEXT
   END IF
   i = i + 1
  WEND
 NEXT
 END SUB
```

Il listato (parte 4; fine)

Il listato

Il programma è costituito da gruppi di subroutines indipendenti, facilmente analizzabili, criticabili (possibilmente in modo costruttivo) e modificabili; è inoltre facilmente traducibile in linguaggi più strutturati data la rigorosa assenza di comandi del tipo GOTO. I sottoprogrammi **Det** e **Inv** sono essenziali solo in fase di ricerca di una o più chiavi idonee.

I fans dell'algebra lineare troveranno comunque il modo di farne un uso proficuo (esempio: un sistema lineare di **n** equazioni in **n** incognite si risolve anche moltiplicando l'inversa della matrice dei coefficienti per il vettore dei termini noti).

Una volta che si disponga di almeno un paio di chiavi idonee, il programma vero e proprio può essere ridotto alle sole routines **Newm:**, **Text:** e **DeCode:**. Le routines di *Load* e **Save** sono state aggiunte solo per comodità: copiare un messaggio cifrato più lungo di 10 caratteri può causare facilmente una crisi di nervi.

Per concludere, un saluto; in codice naturalmente:

```
97U:GW<'0/$y&Es]Ut_2UCK/ (5,4,4,3)
```

...con invito a decifrarlo

di Armando Sforzi

Tre Batch-Files per tutti i gusti

Con i Batch files è possibile sfruttare a fondo le risorse del Sistema Operativo per soddisfare le proprie necessità.

Caratteristica molto interessante è la capacità che AmigaDos possiede di eseguire automaticamente sequenze di comandi: i cosiddetti **Batch File** o **File Script**.

Esempi classici di questi sono le ben note famose sequenze di inizializzazione presenti nella directory **S:** del disco di sistema, tramite le quali (e modificando le quali, naturalmente con criterio) è possibile inizializzare il computer configurandolo, entro certi limiti, in modo diverso a seconda delle esigenze.

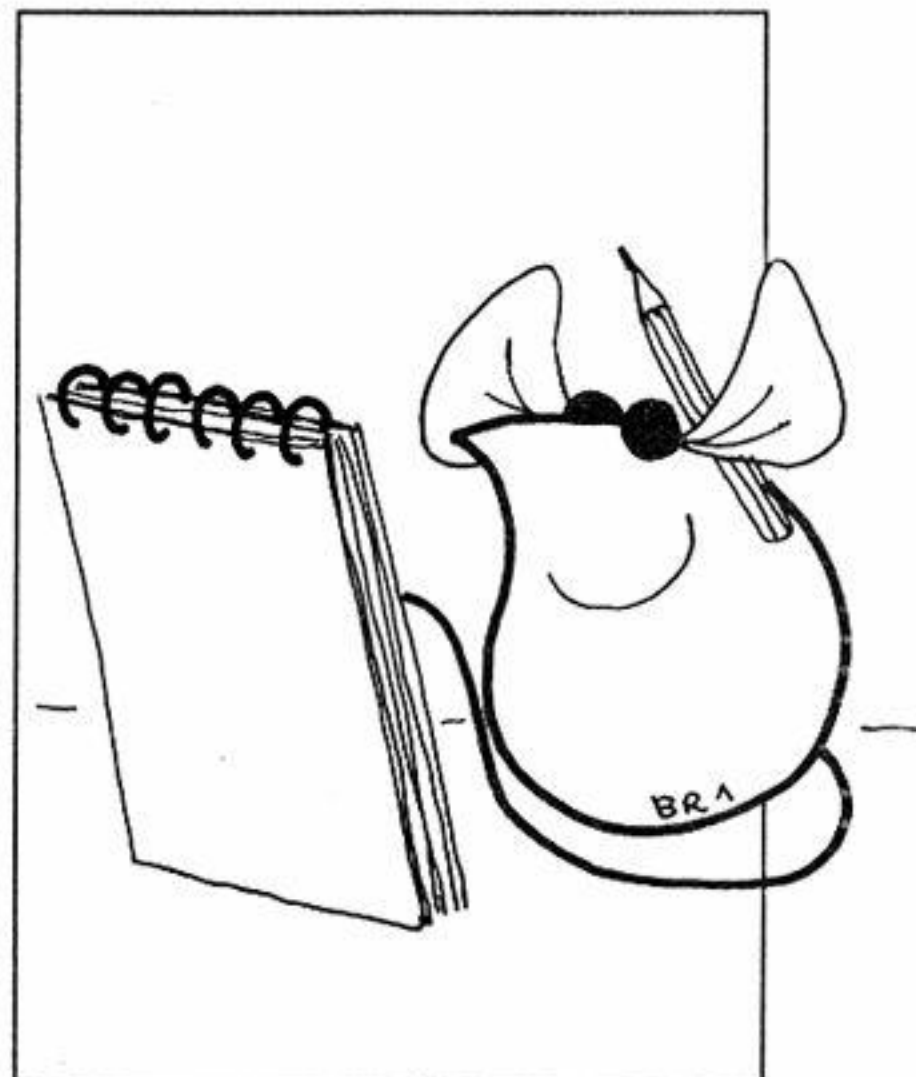
Ma tutto questo non è certamente un novità per coloro che, in qualche modo, abbandonando il rassicurante Workbench e biclickando su **Cli** o, meglio (a partire dalla versione 1.3 del Dos) su **Shell** si sono avventurati in una conoscenza più approfondita del sistema del calcolatore. Una considerazione che forse non tutti possono aver fatto è quella per cui i comandi del Dos, presi nel loro insieme, costituiscono un *vero e proprio linguaggio di programmazione*, permettendo di creare programmi di gestione (i file script, appunto) anche relativamente complessi. Con i files presentati in queste pagine vogliamo, appunto, dare alcune dimostrazioni delle capacità di programmazione del Dos, cercando di perseguire uno scopo didattico non disgiuntamente dall'evidenziare la loro valenza di utilità sfruttabile in applicazioni pratiche.

Il primo batch che presentiamo intende raccogliere la **sfida lanciata su C.C.C n. 79**, dando una risposta concreta alla richiesta di creare una procedura che permetta di calcolare il tempo totale di utilizzo di un certo programma.

Di questo batch, abbondantemente commentato sul listato, analizzeremo ora l'impostazione. Per quanto concerne le caratteristiche delle singole istruzioni, rimandiamo ai numeri di C.C.C. che, a partire dal 75 con la rubrica "Amigafacile", descrivono con estrema chiarezza e completezza i comandi di AmigaDos.

L'idea, per realizzare il **contatore del tempo**, è stata quella di creare un file nel quale siano fissate: l'**ora** in cui si è iniziato a lavorare con un certo programma per la prima volta; l'ora stessa successivamente memorizzata e, di volta in volta, **aggiornata**; l'ora, infine, del **termine** di ciascuna sessione di lavoro.

```
 .key filename/a,filetime/a      ;Preleva gli argomenti
; +++ conta il tempo di lavoro +++
echo "TIME-WORK COUNTER"
copy c:search to ram:
if exists "<filetime>"          ;Se esiste <filetime>...
   search >env:buf "<filetime>" "!" nonum ;cerca "!"
   if not warn                  ;Se la trova...
      copy env:buf to env:date1 ;copia la linea in date1
   endif
else                        ;se no...(la prima volta)
   date >env:date2          ;memorizza la data corrente...
   setenv buf "!"           ;completa di marcatore "!"...
   join env:buf env:date2 to env:date1;per riconoscerla
endif
run >nil: "<filename>"      ;Carica il programma
ask "(Active window and) Press RETURN to end" ;Attende
date >env:date2             ;Memorizzadata finale insieme...
join env:date1 env:date2 to "<filetime>"; ...alla prima
```

Il file Batch da usare in unione con il programma AmigaBasic

```
' Legge il FileTime
' generato dal batch - file
' che conta il tempo di lavoro
DEF FNa (j) = VAL (MID$ (a$, j))
INPUT "FileName"; f$
OPEN f$ FOR INPUT AS 1
a$ = INPUT$ (LOF (1), 1)
CLOSE 1
p1 = INSTR (a$, ": ") - 2
p2 = LEN (a$) - 8
time1 = FNa (p1) * 3600 + FNa (p1 + 3) * 60 + FNa (p1 +
6)
j = FNa (p2): IF j < FNa (p1) THEN j = j + 24
time2 = j * 3600 + FNa (p2 + 3) * 60 + FNa (p2 + 6)
j = time2 - time1
PRINT "HH: MM: SS di lavoro>";
PRINT INT (j / 3600)":";
j = j - INT (j / 3600) * 3600
PRINT INT (j / 60)":";
```

Il listato AmigaBasic che "interpreta" il file Ascii

Per l'utilizzo basta aprire una finestra Cli (o Shell) e lanciare il batch (precedentemente memorizzato su disco col nome che si preferisce) con la sintassi...

`Execute Batch FileName FileTime`

...in cui **Batch** è il nome con cui è memorizzato il nostro file batch, **FileName** è il nome del programma di cui si desidera tener nota del tempo di utilizzo e **FileTime** è il nome del file destinato a contenere le informazioni relative al tempo di utilizzo di FileName.

Naturalmente i tre parametri devono essere eventualmente completi del **Path**, cioè del percorso lungo volumi e directory che il Dos deve fare per rintracciare i file stessi.

Ricordiamo che il comando **Execute** può essere aggirato (apparentemente) settando il flag "s" del file-batch col comando: **`Protect Batch +s`**. In questo modo è possibile attivare il file batch semplicemente richiamandone il nome corredato degli eventuali parametri, come un qualsiasi altro comando Dos.

A sessione di lavoro terminata, si tornerà nella finestra Cli e, dopo averla eventualmente attivata clickando col mouse al suo interno, si premerà il tasto Return, interrompendo così il conteggio del tempo e permettendone la memorizzazione. Per l'interpretazione dei dati del FileTime si è preferito far ricorso ad un banale programmino scritto in AmigaBasic, anch'esso pubblicato in queste pagine.

Un nuovo Locate

Illustriamo ora il secondo file-batch che, in pratica, aggiunge un nuovo comando AmigaDos (implementando l'istruzione di **Locate**), per posizionare dove si vuole, su video, l'output del testo.

E', quindi, un comando del tutto analogo all'omonimo disponibile in AmigaBasic. La sintassi di lancio del nostro brevissimo script è la seguente, avendo avuto cura di settare il flag "**s**" con Protect Batch +s:

`Batch PosY PosX`

La posizione di Home (in alto a sinistra) è individuata da (**1,1**).

Dal punto di vista della programmazione bisogna notare, nel file batch, l'utilizzo

```
.key y/a,x/a                     ;Accetta i parametri
; +++ Locate +++
if <y> gt "0" val                ;Se Y<>0...
   if <x> gt "0" val             ;e X<>0...
      skip exec                  ;esegue il comando
   endif
endif
echo "illegal arguments"         ;Se no segnala errore...
quit                             ;ed esce
lab exec
eval <y> lformat="*E[%NH"        ;Posiziona Y
eval <x> lformat="*E[%NC*E[1D";Posiziona X
```

Il file batch che implementa il comando Locate di AmigaBasic

inconsueto del comando **Eval**, normalmente adoperato (anzi, diciamo pure, scritto appositamente) per valutare semplici espressioni numeriche intere. In questo caso, invece, Eval è chiamato a svolgere un compito completamente diverso: preleva il valore del parametro associato ad execute, quindi, tramite l'opzione **Lformat** lo inserisce, al posto di "**%N**", in una stringa contenente i caratteri di controllo del cursore, infine stampa la stringa. Il tutto in una sola linea, semplificando quindi il programma e rendendolo più veloce. Cogliamo l'occasione, anzi, per far notare come, sfruttando con un pizzico di creatività le potenzialità dei comandi, sia talvolta possibile ottenere effetti, per così dire, non canonici, che permettono di raggiungere gli obiettivi più velocemente che non percorrendo un normale itinerario di programmazione.

```
.key file/a,op/a,arg1,arg2,arg3  ;Accetta i parametri
; +++ ricerca condizionata +++
copy c:search to ram:        ;Copia in ram ...
copy c:if to ram:            ;alcuni comandi...
copy c:endif to ram:         ;per accelerare...
copy c:copy to ram:          ;il processo...
copy c:getenv to ram:        ;di ricerca...
copy c:skip to ram:          ;
copy c:lab to ram:           ;
if "<op>" eq "and"           ;Se la ricerca e' tipo "and"...
   skip AndSection           ;salta alla sezione giusta...
else                         ;altrimenti...
   if not "<op>" eq "or"     ;se non e' nemmeno tipo 'or'...
      echo "bad arguments"   ;segnala errore...
      quit                          ;ed esce
   endif
endif
lab OrSection                       ;*** RICERCA "OR" ***
if not "<arg1>" eq ""               ;Se esiste argomento
 search >env:buf1 "<file>" "<arg1>" nonum;ricorrenze...
   if not warn                      ;se ha trovato qualcosa.
      skip end                      ;salta alla fine
   endif
endif
if not "<arg2>" eq ""               ;...come sopra...
   search >env:buf1 "<file>" "<arg2>" nonum;
   if not warn                      ;
      skip end                      ;
   endif
endif
if not "<arg3>" eq ""               ;...come sopra...
   search >env:buf1 "<file>" "<arg3>" nonum;
endif                               ;
skip end                            ;
lab AndSection                      ;*** RICERCA "AND" ***
if not "<arg1>" eq ""               ;Se esiste l'argomento
   search >env:buf1 "<file>" "<arg1>" nonum;cerca ricorr.
   copy env:buf1 env:buf2           ;e le copia nel buffer
endif
if not "<arg2>" eq ""               ;...come sopra...
   search >env:buf1 env:buf2 "<arg2>" nonum;
   copy env:buf1 env:buf2           ;
endif
if not "<arg3>" eq ""               ;...come sopra...
   search >env:buf1 env:buf2 "<arg3>" nonum;
endif
lab end                             ;*** fine ***
getenv buf1                         ;Stampa dei risultati
```

L'ultimo batch file consente ricerche di tipo Or e And

L'ultimo batch

Eccoci, infine, a descrivere il terzo ed ultimo file-batch, il più lungo e, forse, il più utile. Esso, infatti, permette di effettuare una ricerca di tipo "**Or**" oppure "**And**" tra le linee di un file.

E' bene ricordare che una **linea** è un segmento di caratteri Ascii terminante col carattere LF (**$0A**).

Non ci sono particolari tecniche di programmazione da evidenziare se non quella lunga teoria di **Copy** iniziale, che serve per copiare in RAM i comandi del Dos che compongono la sezione del programma dedicata alla ricerca, in modo da rendere molto veloci le operazioni. La sintassi è (a patto, lo ripetiamo ancora, di aver settato, come al solito, il flag "**s**" col comando: Protect Batch +s)...

```
Batch File Op [arg1] [arg2]
[arg3]
```

...in cui **Batch** è il nome assegnato al nostro file, che potrebbe essere ad esempio *Research*; **File** è il nome del file in cui effettuare la ricerca; **Op** indica il tipo di ricerca: bisogna scrivere quindi "**or**" oppure "**and**".

Questo argomento, come ovviamente il precedente, è obbligatorio, altrimenti il programma segnalerà errore. I tre parametri che seguono tra parentesi quadre sono le chiavi di ricerca e non necessariamente devono essere passati tutti e tre.

In pratica, se si effettua una ricerca di tipo "or", il programma cercherà e stamperà tutte quelle linee che contengono **arg1** oppure **arg2** oppure **arg3**, mentre avviando una ricerca di tipo "**and**" le linee evidenziate saranno eventualmente quelle che conterranno arg1 "**e**" arg2 "**e**" arg3. Questo file-batch, come abbiamo già detto, è utile perchè può essere inserito in un programma, sempre di tipo batch, che gestisca, ad esempio, un database.

SIRONI GD&P

Una pelliccia ancora viva.

Animal Amnesty è una associazione per i diritti degli animali che agisce soprattutto attraverso l'informazione.
Propone ai suoi soci dei facili suggerimenti pratici e concreti per aiutare a difendere gli animali.

Se vuoi essere tenuto informato e collaborare anche tu per un mondo con meno sofferenza (umana e non umana), associati ad **Animal Amnesty**.

Per iscriverti è sufficiente versare la quota prescelta sul ccp 11396207 intestato ad **Animal Amnesty** / Galleria Passerella, 1 20122 Milano, specificando sul retro della cedola la causale. Le quote sono:

L. 15.000 Giovane (fino a 18 anni o disoccupato)
L. 30.000 Aderente
L. 60.000 Sostenitore
L. Donazione
Insieme saremo più efficaci se più grande sarà il contributo.

ANIMAL AMNESTY

Quanti animali per una pelliccia?
10/24 Volpi
30/70 Visoni
130/200 Cincillà
16/20 Castori
100/400 Scoiattoli
180/240 Ermellini
30/45 Agnelli Broadtail
20/30 Procioni
26/34 Nutrie (Castorino)
8/12 Linci

di Luigi Callegari

Vettori, puntatori e matrici

Il "puntatore" è stato incontrato in precedenti chiacchierate sul C; è il momento di approfondire l'argomento

Trattandosi di un concetto, ad un tempo, complesso e cardinale nell'economia dei programmi scritti in C, è fondamentale capirlo bene dedicandovi parecchio spazio.

Oltretutto, il **puntatore** è alla base della gestione dei vettori, delle matrici (in senso stretto, un vettore è già una matrice monodimensionale) e, come già detto, anche delle stringhe alfanumeriche, delle parole insomma, che sono considerate e manipolate dai programmi in C come vettori di singoli caratteri.

Puntatori e memoria

Quando si scrive un programma si ha spesso bisogno di gestire bytes di memoria.

Per farlo è necessario che gli stessi siano identificabili dall'**indirizzo**, valore numerico che identifica nella mappa della memoria di Amiga (che si estende normalmente da 0 a 8,388,608, per un totale di circa 8 milioni, 8 megabyte, di singole locazioni).

Mentre le variabili sono gestite automaticamente dal compilatore, dal punto di vista del programmatore, le locazioni di memoria devono essere indirizzate tramite puntatori.

In effetti, il codice compilato gestisce ciò che noi chiamiamo variabili (e usiamo nel nostro testo) mediante puntatori, che indirizzano (indicano, o "puntano" appunto) i valori contenuti in locazioni di memoria; tutto ciò avviene comunque in modo a noi trasparente.

Operatori

La stretta relazione che lega variabili a puntatori ci porta a scoprire l'operatore "**indirizzo**".

Questo operatore, ovvero simbolo che svolge una particolare operazione (come "+" è l'operatore di somma matematica), svolge la funzione opposta del puntatore, in effetti, dato che consente di ricavare l'indirizzo di memoria a cui sono allocati i dati relativi ad una variabile. La simbologia del linguaggio C è come segue:

* Simbolo di puntatore
& Simbolo di indirizzo

Questi simboli devono essere associati normalmente, salvo alcune eccezioni relative all'operatore puntatore, ad un identificatore.

Ad esempio, l'esecuzione delle seguenti linee...

```
p = &x
y = *p
```

...eseguono due funzioni complementari. La prima assegna alla variabile "**p**" il valore assoluto corrispondente all'indirizzo (**&**) di memoria a cui è memorizzala la variabile "**x**". La seconda linea dice invece di assegnare alla variabile **y** il valore che è memorizzato a partire dall'indirizzo puntato (*****) dalla variabile **p**, che nella

```
/* Scambio dei valori x et y,
in modo errato e corretto */

void cor(), err();

void main()
{
  int x,y;

  printf("Inserisci x et y\n\n");
  scanf("%d %d", &x, &y );
  printf("\n\nx=%d y=%d", x,y );

  err( x,y ); /* Errato */

  printf("\n\nModo errato:\n\n");
  printf("x=%d y=%d", x,y );

  cor( &x, &y );
  printf("\n\nModo corretto:\n\n");
  printf("x=%d, y=%d", x,y );
  printf("\n\nFine lavoro\n\n");
}

void err( int a, int b )
{
  int t;

  t = a; a = b; b = t;
}

void cor( int *pa, int *pb )
{
  int t;
  t = *pa;
  *pa = *pb;
  *pb = t;
}
```

Figura 1

fattispecie si chiama appunto impropriamente "puntatore".

In pratica, le due linee precedenti sono l'equivalente di una qualunque delle due linee seguenti:

```
y = *( &x )
y = x
```

Probabilmente sono bastate queste poche righe a confondere le idee di molti, non tanto per la difficoltà dei concetti originali di puntatore ed indirizzo, bensì per il fatto che anche i puntatori e gli indirizzi sono assegnabili a variabili.

L'importante è sapere distinguere mentalmente le variabili di tipo generico (dichiarate come **short, int, long** eccetera) dalle variabili puntatori. Materialmente, una **variabile puntatore** è un gruppo di locazioni di memoria sufficienti a contenere qualunque indirizzo nella mappa di memoria. Nel caso di Amiga, una variabile puntatore occupa pertanto **32 bit**. Infatti, solo variabili che possono contenere un **qualunque** indirizzo di memoria possono essere utili come puntatori.

La distinzione tra puntatori e variabili deve essere specificata al momento della definizione del tipo di contenuto delle variabili.

Come visto poc'anzi, il risultato pratico derivante dall'esecuzione delle due linee dell'esempio e' quello di una assegnazione diretta "**y=x**". Infatti, ad y viene assegnato nella prima linea il valore contenuto nella locazione di memoria a cui punta &x, il quale rappresenta l'indirizzo di memoria a partire da cui è memorizzato il contenuto di X.

In pratica, il valore contenuto dal contenuto della locazione di memoria il cui puntatore è pari all'indirizzo da cui è memorizzato **x** corrisponde effettivamente al valore x; perciò il contenuto di **y** sarà uguale al contenuto di "x". Semplice, no?

Notare che abbiamo detto *"a partire da cui"*: un puntatore, infatti, contiene materialmente la prima locazione di memoria in cui è conservato un dato che, a seconda del tipo, può risiedere in più locazioni consecutive. Ad esempio, nel caso di programmi compilati con il **SAS/C**, una variabile Long occupa 32 bit (2 word consecutive).

Definizione

La definizione di un puntatore ha un duplice effetto: definire il tipo di valori a cui punta e definire la dimensione della variabile che conterrà il valore del puntatore; perciò la seguente linea...

```
char *p;
```

...comunica al compilatore che i valori contenuti in **p** (a 32 bit, come long), rappresentano un puntatore a dati di tipo **char** (carattere). Benchè si parli materialmente, per i motivi di allocazione a 32 bit detti prima, sempre di valori di tipo **intero lúngo**, assegneremo ad una variabile puntatore a carattere solo una variabile dello stesso tipo. In altre parole, il compilatore controllerà che a questo puntato si assegni sempre l'indirizzo di un carattere e lo gestirà, come vedremo tra poco, secondo regole adatte.

Calcoli

E' possibile effettuare calcoli aritmetici tra variabili puntatori e variabili di **tipo long**. Ad esempio, nel programma di figura 1, si può vedere uno scambio tra variabili effettuato da una funzione richia-

```
/* Visualizzazione contenuto della
memoria in hex ed ASCII partendo
da indirizzo specificato */

void visli( unsigned char *i )
{
    int c,d;

    printf("%08x:", i );
    for ( c = 1 ; c <= 16 ; c++ ) {
        d = ( *i>=0 ? *i : (256 + *i));
        printf("%02x ", d );
        ++i;
        }

    i -= 16;
    printf( "- ");
    for ( c=1 ; c<= 16; c++ )
    {
        d = ( *i > 31 && *i<128 ) ? *i : '.';
        printf( "%c", d );
        ++i;
        }
    printf("\n");
}

void main()
{
    char f, *x;
    short b;
    printf("\n\nInserisci indirizzo esadecimale: ");
    scanf("%8x", &x );
    printf("\n\n\n");
    f = 0;
    while( f != 'f' )
    {
        for ( b = 0 ; b<18 ; b++ ) {
            visli( x );
            x += 16;
            }
        printf("\n\nEnter per continuare,");
        printf("f per finire  ");
        scanf( "%c", &f );
        printf("\n\n");
    }
}
```

Figura 2

```
/* Definizione di matrice di 2
puntatori e assegnazione di
stringhe e numero di elementi */

void main( void )
{

long x;
static char *giorni[] = {
    "inesistente",
    "lunedì,
    "martedì,
    "mercoledì,
    "giovedì,
    "venerdì,
    "sabato",
    "domenica"
};
    printf("\n\nInserisci giorno");
    printf(" della settimana (1.7): ");
    scanf("%d",&x);

    if ( x>0 && x<8 ) {
        printf("\n\nIl %d' giorno ", x );
        printf("della settimana : ");
        }


    else {
        x = 0;
        printf("\n\nGiorno ");
        }
    printf("%s\n\n", giorni[x]);
}
```

Fig. 3

mata: sono inclusi sia un metodo errato, sia un metodo corretto.

Nell'esempio, vogliamo passare alla funzione chiamata un paio di parametri: nella funzione **err()** questi sono il contenuto di **x** e quello di **y**, mentre nella funzione **cor()** vengono passati gli indirizzi di **x** ed **y**, rispettivamente.

Dal tipo di elaborazione effettuata dal programma possiamo notare che la funzione err() scambia solo apparentemente i valori x ed y. O meglio, scambia i valori solo al proprio interno, dato che non restituisce i valori cambiati al programma chiamante.

In altri linguaggi, come il Pascal, si direbbe che la funzione viene richiamata passando i parametri per valore, mentre per modificare i valori agli occhi di tutte le funzioni del programma ("globalmente") si devono passare i parametri per nome.

Ritornando al linguaggio C, per effettuare lo scambio corretto dovremo utilizzare il secondo metodo, lavorando con i puntatori alle variabili. La variabile **t** serve soltanto come mezzo di transizione, come punto di appoggio, per scambiare le variabili senza perdere alcuna delle due.

Con una definizione del tipo...

```
long *x;
```

...accadrà che per tutte le operazioni del tipo...

```
a = *(x+1)
```

...ad esempio, otterremo un calcolo scalare dell'indirizzo a cui puntare. Vale a dire, se **x** punta alla locazione 1000, in cui si trova un valore "long" che è lungo 4 byte, allora (x+1) punterà implicitamente alla locazione 1004. Questo tipo di calcoli è del tutto trasparente per l'utente, ma è importante, ovviamente, sapere che il compilatore li svolge.

Vettori

I vettori di dati, numerici od alfanumerici, sono definiti mediante un identificatore ("nome") e le parentesi quadre. L'**identificatore**, ovvero il nome del vettore, è in realtà un puntatore contenente l'indirizzo di partenza dei dati appartenenti al vettore stesso. Per esempio, la stringa definita con...

```
char nome[10];
```

...indica un vettore di 10 variabili carattere, associate all'identificatore "**nome**". Quest'ultimo, preso singolarmente, non è altro che il puntatore indirizzante la locazione di memoria della quale è memorizzata la sequenza di caratteri. Quindi, il contenuto di "nome" risulterà identico a `&nome[0]`, dove lo zero indica il primo elemento della sequenza del vettore, quindi è possibile usarlo ad esempio in **printf()** o **scanf()** per indicare l'indirizzo di inizio della stringa da visualizzare o inserire.

La differenza che intercorre tra definizione di un puntatore e definizione di un vettore consiste nel fatto che la prima predispone una variabile. Ovvero, con quest'ultima sono possibili operazioni aritmetiche, mentre la seconda definisce un valore costante al quale viene sommato, in modo scalare, il valore posto tra le parentesi quadre.

Inoltre la definizione di un vettore crea lo spazio in memoria atto a contenere tutti i dati, mentre il puntatore è solo una variabile. In base a quanto detto sono ovviamente consentite le seguenti dichiarazioni:

```
char nome[20], *punt, x;
punt = &nome[0];
punt = nome;
x = nome[5];
x = *(punt+5);
x = punt[5];
```

...dove il valore **punt** assume sempre lo stesso significato, mentre alla variabile **x** viene assegnato lo stesso valore. E' invece **errata** la specifica:

```
punt = &( x+ 2 );
```

Infatti, la "x" è disposta in memoria: posto a 2 il suo valore sarebbe come chiedere l'indirizzo di 2+2, che essendo una costante numerica non può essere indirizzata, cioè non se ne può calcolare l'indirizzo perchè non avrebbe senso.

Un esempio

Vediamo ora un altro esempio con i puntatori nel listato di figura 2, programma che consente la visualizzazione del contenuto della memoria a partire dall'indirizzo battuto dall'utente.

I valori riportati in ogni linea indicano: l'**indirizzo** del primo byte, il **contenuto** in formato esadecimale sottoforma di 16 byte consecutivi e gli stessi 16 byte in formato ASCII, con i codici non stampabili sostituiti da un punto.

```
/* Ricerca sequenza caratteri
entro una stringa. Ritorna il
carattere di partenza se OK,
zero altrimenti */

#include <stdio.h>

long instr();

void main()
{
    long n;
    char s1[100], s2[100];

    printf("\n\nInserisci stringa:\n");
    scanf("%s", s1 );

    printf("\nInserisci la sequenza ");
    printf("di caratteri da cercare: ");
    scanf( "%s", s2 );

    n = instr( s1, s2 );
    printf( "\nStringa 1: %s", s1 );
    printf( "\nStringa 2: %s\n\n", s2 );
    if ( n == 0 )
        printf( "Sequenza non trovata\n\n");
    else {
        printf("La sequenza parte dal ");
        printf("carattere n.%d\n\n\n",n );
        }
}

long instr( char *d1, char *d2 )
{
    char *x, *y, *a, f;
    x = d1, f = 0;

    for ( x = d1; f == 0 && *x != '\0' ; x++ ) {
        a = x, f = 1, y = d2;

        while( f > 0 && *y != '\0') {
            f = (*a != '\0' && *a == *y ) ? 1 : 0;
            ++a;
            ++y;
            }
        }
    return( f==1 ? x-d1 : 0 );
}
```

Fig. 4

La funzione **visli()** visualizza una linea di dati a partire da un indirizzo specificato, definito con il passaggio del puntatore x. Si può notare che all'interno della funzione il puntatore **i**, che indirizza il dato corrente, è utilizzato sia come variabile (**i++**) sia in luogo del dato a cui punta (***i**, ***i0** eccetera). Nel blocco del primo ciclo della visli() l'assegnazione di **d** include un operatore condizionale che restituisce ***i**, purchè questo non sia negativo.

Matrici

Una matrice di dati è materialmente formata da un'area di memoria consecutiva gestita per righe e colonne, in modo comunque trasparente all'utente.

Per manipolare una matrice possiamo pensarla anche come vettore di puntatori, ovvero una serie di puntatori associati all'inizio di ogni linea di una matrice. Definendo, ad esempio,...

```
char mat[10][10], *punt[10];
```

...otterremo sia una matrice di 10 per 10 caratteri, sia un vettore di 10 puntatori a carattere. Nel primo caso il compilatore allocherà materialmente spazio per cento caratteri, nel secondo soltanto lo spazio per dieci puntatori.

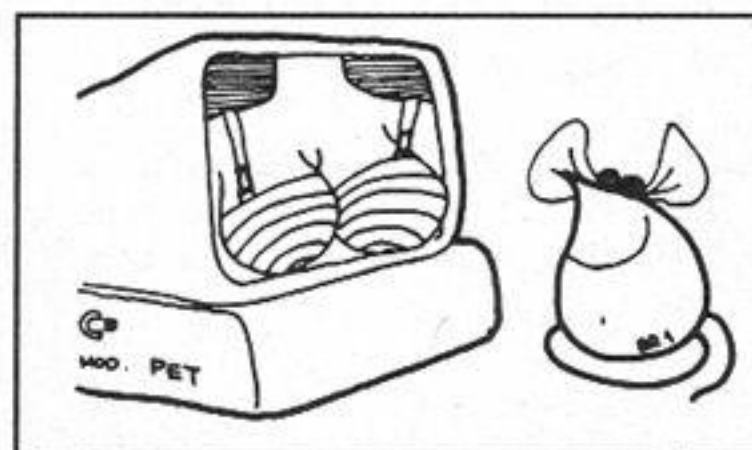
Nel listato di figura 3 vediamo, appunto, un esempio di programma che definisce un vettore di puntatori e contemporaneamente assegna delle stringhe.

Si noti che tra le parentesi quadre del vettore **"giorni"** non è specificata la lunghezza, infatti la sua definizione viene effettuata automaticamente durante l'assegnazione. L'indirizzo del primo carattere di ogni elemento stringa definito in memoria verrà assegnato ai puntatori del vettore di puntatori.

Quindi il puntatore **giorni[0]** identificher' la stringa "inesistente", **giorni[1]** punterà a lunedì e così via per gli otto elementi del vettore. La visualizzazione della stringa selezionata in base al valore inserito viene effettuata passando alla funzione **printf()** il puntatore **giorni[x]**.

Nel listato di figura 4 possiamo comprendere meglio l'applicazione pratica dei vettori e dei puntatori. Si tratta di una routine che cerca una sequenza di caratteri all'interno di una stringa. La **main()** chiama la funzione **instr()** che si occupa di effettuare la ricerca. I parametri necessari sono: puntatore alla stringa base (**d1**) e puntatore alla sequenza di caratteri da cercare (**d2**). Il confronto inizia dal primo carattere della stringa e procede sequenzialmente sino al termine della stessa. Se viene trovata l'eguaglianza tra i primi caratteri si prosegue con i rimanenti della stringa da scandire. Se il confronto è positivo per tutti i caratteri, la funzione restituisce un valore long che punta al primo carattere della stringa, altrimenti viene restituito un valore zero.

Conclusioni

La prossima volta inizieremo a parlare dell'uso delle routine del sistema operativo di Amiga da linguaggio C, introducendo discorsi non più riconducibili allo standard ANSI C, ma peculiari ed esclusivi per Amiga.

Il meglio di C.C.C.

Per la gioia dei nostri lettori, pubblicheremo, periodicamente, un "indice" degli argomenti (di recente pubblicazione) che riteniamo di maggior interesse per gli utenti della nostra rivista. La suddivisione per argomenti faciliterà la ricerca degli articoli; il nome di questi, riportato in questa pagina, non sempre corrisponde a quello originale. Ciò per far meglio comprendere l'argomento trattato. Il numero in neretto, ovviamente, corrisponde al numero del fascicolo della rivista.

SYSTEMS EDITORIALE PER TE

La voce
Aggiunge al C/64 nuovi comandi Basic che consentono sia di far parlare il computer, sia di farlo Cantare! Diversi esempi allegati.
Cassetta: L. 12000 - Disco: L. 15000

Raffaello
Un programma completo per disegnare, a colori, con il C/64: linee, cerchi, quadrati, eccetera. Valido sia per disegno a mano libera che geometrico.
Cassetta: L. 10000

Oroscopo
Devi solo digitare la data di nascita e le coordinate geografiche del luogo che ti ha dato i natali. Vengono quindi elaborate le varie informazioni (case, influenze dei segni astrali, eccetera) e visualizzato un profilo del tuo carattere. Valido per qualsiasi anno, è indicato sia agli esperti sia ai meno introdotti. E' allegata una tabella delle coordinate delle più note città italiane e l'elenco delle ore legali in Italia dal 1916 al 1978.
Cassetta: L. 12000 - Disco: L. 12000

Computer Music
Cassetta contenente numerosi brani di successo da far eseguire, in interrupt, al tuo C/64 sfruttando, fino in fondo, il suo generatore sonoro (SID).
Cassetta: L. 12000

Gestione Familiare
Il più noto ed economico programma per controllare le spese e i guadagni di una famiglia.
Cassetta: L. 10000 - Disco: L. 10000

Banca Dati
Il più noto ed economico programma per gestire dati di qualsiasi natura.
Cassetta: L. 10000 - Disco: L. 10000

Matematica finanziaria
Un programma completo per la soluzione dei più frequenti problemi del settore.
Cassetta: L. 10000 - Disco: L. 20000

Analisi di bilancio
Uno strumento efficace per determinare con precisione i calcoli necessari ad un corretto bilancio.
Cassetta: L. 10000 - Disco: L. 20000

Corso di Basic
Confezione contenente quattro cassette per imparare velocemente le caratteristiche delle istruzioni Basic del C/64 e i rudimenti di programmazione. Interattivo.
Cassetta: L. 19000

Corso di Assembler
Un corso completo su cassetta per chi ha deciso di abbandonare il Basic del C/64 per addentrarsi nello studio delle potenzialità del microprocessore 6502. Interattivo.
Cassetta: L. 10000

Logo Systems
Il linguaggio più facile ed intuitivo esistente nel campo dell'informatica; ideale per far avvicinare i bambini al calcolatore.
Diversi esempi allegati.
Cassetta: L. 6500

Compilatore Grafico Matematico
Uno straordinario programma compilatore, di uso semplicissimo, che permette di tracciare, sul C/64, grafici matematici Hi-Res ad altissima velocità. Esempi d'uso allegati.
Cassetta: L. 8000

Emulatore Ms-Dos e Gw-Basic
Un prodotto, unico nel suo genere, che permette di usare, sul C/64 dotato di drive, la sintassi tipica del più diffuso sistema operativo del mondo. Ideale per studenti.
Solo su disco: L. 20000

Emulatore Turbo Pascal 64
Permette di usare le più importanti forme sintattiche del linguaggio Turbo Pascal (anche grafiche!) usando un semplice C/64 dotato di drive.
Ideale per studenti.
Disco: L. 19000

Speciale drive
Questo speciale fascicolo costituisce una guida di riferimento per le unità a disco del C64/128.
Comprende anche un velocissimo turbo-disk più la mappa completa della memoria del drive.
Fascicolo + disco: L. 12000

Utility 1
Un dischetto pieno zeppo di programmi speciali per chi opera frequentemente con il drive.
Disco: L. 12000

Utility 2
Seconda raccolta di utility indispensabili per realizzare sofisticate procedure di programmazione.
Disco: L. 15000

Graphic Expander 128
Per usare il C/128 (in modo 128 e su 80 colonne) in modo grafico Hi-res. Aggiunge nuove, potenti istruzioni Basic per disegnare in Hi-Res con la massima velocità in modalità 80 colonne.
Disco: L. 27000

Directory
Come è noto, a partire dal N. 10 di "Software Club" (la rivista su disco per l'utente dei "piccoli" computer Commodore), vengono riportati tutti i listati, in formato C/64-C/128, pubblicati su "Commodore Computer Club".
In precedenza tali listati venivano inseriti, mensilmente, in un dischetto, di nome "Directory", che oltre ai programmi di C.C.C. ospitava decine di altri file tra cui musiche nell'interrupt, giochi, listati inviati dai lettori e altro.
Ogni disco, dal prezzo irrisorio, contiene quindi una vera miniera di software. Ordinando i dischetti di "Directory" si tenga conto che al N. 1 corrispondeva il contenuto del N. 34 di "Commodore Computer Club", al N. 2 il N. 35 e così via.
Ogni dischetto: L. 10000

Super Tot '64
La nuova e completa edizione del programma Tot 13 con tutti i sistemi di riduzione e di condizionamento.
Ampia sezione dedicata alla teoria.
fascicolo + disco: L. 15000

Amiga Totospeed
Finalmente anche per Amiga un programma orientato alla compilazione delle schedine totocalcio.
Fai tredici conil tuo Amiga.
disco: L. 20000

SYSTEMS EDITORIALE PER TE

Disk'o'teca
Grazie a questa nutrita raccolta di brani musicali potrete divertirvi ascoltando i migliori brani prodotti dai vostri beniamini, oltre a una serie di composizioni prodotte "in casa".
In omaggio un bellissimo poster di Sting.
Disco: L.15.000

Assaggio di primavera
Esclusivo!
In un'unica confezione potrete trovare ben due cassette di videogiochi assieme a un comodo e funzionale joystick.
Cassette: L. 15.000

LIBRI TASCABILI

64 programmi per il C/64
Raccolta di programmi (giochi e utilità) semplici da digitare e da usare. Ideale per i principianti. **(126 pag.)**
L. 4800

I miei amici C/16 e Plus/4
Il volumetto, di facile apprendimento, rappresenta un vero e proprio mini-corso di Basic per i due computer Commodore. Numerosi programmi, di immediata digitazione, completano la parte teorica. **(127 pag.)**
L. 7000

62 programmi per C/16, Plus/4
Raccolta di numerosi programmi, molto brevi e semplici da digitare, per conoscere più a fondo il proprio elaboratore.
Ideale per i principianti. **(127 pag.)**
L. 6500

Micro Pascal 64
Descrizione accurata della sintassi usata dal linguaggio Pascal "classico". Completa il volume un programma di emulazione del PL/O sia in formato Microsoft sia in versione C/64 (da chiedere, a parte, su disco). **(125 pag.)**
L. 7000

Dal registratore al Drive
Esame accurato delle istruzioni relative alle due più popolari periferiche del C/64. Diversi programmi applicativi ed esempi d'uso. (94 pag.)
L. 7000

Il linguaggio Pascal
Esame approfondito della sintassi usata nel famoso compilatore. (112 pag.)
L. 5000

Simulazioni e test per la didattica
Raccolta di numerosi programmi che approfondiscono e tendono a completare la trattazione già affrontata sul precedente volume. **(127 pag.)**
L. 7000

Dizionario dell'Informatica
Dizionario inglese-italiano di tutti i termini usati nell'informatica. (Edizione completa). **(385 pag.)**
L. 10000

Word processing: istruzioni per l'uso
Raccolta delle principali istruzioni dei più diffusi programmi di w/p per i sistemi Ms-Dos: Word-Star, Samna, Multimate Advantage, Word 3. **(79 pag.)**
L. 5000

Unix
Un volumetto per saperne di più sul sistema operativo professionale per eccellenza.
Un necessario compendio per l'utente sia avanzato che inesperto **(91 pag.)**
L. 5000

ABBONAMENTO

Commodore Computer Club
11 fascicoli: L. 60.000

ARRETRATI

Ciascun numero arretrato di C.C.C. L. 6.000

Come richiedere i prodotti Systems

Coloro che desiderano procurarsi i prodotti della Systems Editoriale devono inviare, oltre alla cifra risultante dalla somma dei singoli prodotti, L. 3500 per spese di imballo e spedizione, oppure L. 6000 se si desidera la spedizione per mezzo raccomandata.
Le spese di imballo e spedizione sono a carico della Systems se ciascun ordine è pari ad almeno L. 50000.
Per gli ordini, compilare un normale modulo di C/C postale indirizzato a:

C/C Postale N. 37 95 22 07
Systems Editoriale Srl
Via Mosè, 22
20090 Opera (MI)

Non dimenticate di indicare chiaramente, sul retro del modulo (nello spazio indicato con "Causale del versamento"), non solo il vostro nominativo completo di recapito telefonico, ma anche i prodotti desiderati ed il tipo di spedizione da effettuare.
Per sveltire la procedura di spedizione sarebbe opportuno inviare, a parte, una lettera riassuntiva dell'ordine effettuato, allegando una fotocopia della ricevuta del versamento.
Chi volesse ricevere più celermente la confezione deve inviare la somma richiesta mediante assegno circolare, oppure normale assegno bancario (non trasferibile o barrato due volte) intestato a:

Systems Editoriale
Milano